目　录

基　础　篇

应　用　篇

实 验 篇


基础篇

第 1 章　MATLAB 概述

自 20 世纪 80 年代以来，出现了科学计算语言，亦称数学软件。比较流行的有 MATLAB、Mathematica、Mathcad、Maple 等，因为它们具有功能强、效率高、简单易学等特点，使其在许多领域得到广泛应用。目前流行的几种科学计算软件各具特点，而且都在不断地发展，新的版本不断出现，但就影响而言，影响最大、流行最广的当属 MATLAB 语言。

本章先介绍 MATLAB 的发展和主要功能，然后介绍 MATLAB 软件系统的使用，最后通过几个例子演示 MATLAB 的功能。通过本章的学习，读者将对 MATLAB 语言许多诱人的特点有一个感性认识，为今后的学习奠定基础。

1.1　MATLAB 的发展

MATLAB 是英文 MATrix LABoratory(矩阵实验室)的缩写。1980 年前后，时任美国新墨西哥大学计算机科学系主任的 Cleve Moler 教授在给学生讲授线性代数课程时，想教学生使用当时流行的线性代数软件包 LINPACK 和基于特征值计算的软件包 EISPACK，但发现许多高级语言调用 LINPACK 和 EISPACK 软件包极为不便，于是，Cleve Moler 教授便着手编写了接口程序并命名为 MATLAB，这便是 MATLAB 的雏形。

早期的 MATLAB 是用 FORTRAN 语言编写的，尽管功能十分简单，但由于是免费软件，还是吸引了大批使用者。经过几年的校际流传，在 John Little 的推动下，由 John Little、Cleve Moler 和 Steve Bangert 合作，于 1984 年成立了 MathWorks 公司，并正式推出 MATLAB 第 1 版(DOS 版)。从这时起，MATLAB 的核心采用 C 语言编写，功能也越来越强。它不仅具有数值计算功能，而且还具有符号计算、图形处理等功能。

以后，MATLAB 版本不断更新。MathWorks 公司于 1992 年推出了具有划时代意义的 4.0 版，并于 1993 年推出了其微机版，该版本可以配合 Windows 3.x 一起使用，使其应用范围越来越广。1994 年推出的 4.2 版扩充了 4.0 版的功能，尤其在图形界面设计方面提供了新的方法。1997 年推出的 5.0 版提供了更多的数据结构，如结构数据、单元数据、多维矩阵、对象与类等，使编程更方便。1999 年初推出的 5.3 版在很多方面又进行了进一步改进。2001 年 7 月，Math Works 公司推出了 MATLAB 的最新版本 6.1 版，6.1 版对计算机的配置要求比较高，很多用户仍使用 5.3 版，因此本书以 5.3 版为基础，全面介绍 MATLAB 的功能与使用技巧。

目前，MATLAB 已经不仅仅是一个“矩阵实验室”了，它已成为一种广泛应用于工程计算及数值分析领域的新型高级语言。MATLAB 功能强大、简单易学、编程效率高，因而深受广大科技工作者的欢迎。在欧美各高等院校，MATLAB 已成为线性代数、自动控制理论、数字信号处理、时间序列分析、动态系统仿真、图像处理等课程的基本教学工具，成为理工科本科生、硕士生以及博士生必须掌握的基本编程语言。在科研和工业生产领域，MATLAB 已被广泛地用于研究和解决各

种具体的工程问题。近年来，MATLAB 在我国也开始流行，应用 MATLAB 的单位和个人急剧增加。可以预见，MATLAB 将在我国科学研究和工程应用中发挥越来越大的作用。

1.2 MATLAB 的主要功能

MATLAB 自 1984 年由 MathWorks 公司推向市场以来，历经十几年的发展和改进，现已逐步风靡世界。其可靠的数值计算和符号计算功能、强大的绘图功能、简单易学的语言体系以及为数众多的应用工具箱使其在科技应用软件中备受瞩目。

1. 数值计算和符号计算功能

科学计算有数值计算和符号计算之分。MATLAB 的数值计算功能非常强大，它提供了十分丰富的数值计算函数，而且所采用的数值计算算法都是国际公认的、最先进的、可靠的算法，其程序由世界一流专家编制，并经高度优化。高质量的数值计算功能为 MATLAB 赢得了声誉。

在实际应用中，除了数值计算外，在符号计算领域往往要得到问题的解析解，MATLAB 和著名的符号计算语言 Maple 结合，实现了 MATLAB 的符号计算功能。

2. 绘图功能

MATLAB 提供了两个层次的绘图操作：一种是对图形句柄进行的低层绘图操作，另一种是建立在低层绘图操作之上的高层绘图操作。利用 MATLAB 的高层绘图操作可以轻而易举地绘制各种图形。利用 MATLAB 图形句柄操作，可以随心所欲地对图形元素进行各种操作，为用户在图形表现方面开拓了一个广阔的、没有丝毫束缚的空间。

3. MATLAB 语言体系

MATLAB 是一种高级的科学计算语言。它具有程序结构控制、函数调用、数据结构、输入输出、面向对象等程序语言特征。使用 MATLAB 可以很容易地实现 BASIC、FORTRAN、C 等传统语言的几乎全部功能，包括 Windows 图形用户界面的设计，而且简单易学，编程效率高。因此，对于从事数值计算、计算机辅助设计和系统仿真等领域的人员来说，用 MATLAB 编程的确是一个最佳选择。

MATLAB 是解释性语言，程序执行速度较慢，而且不能脱离 MATLAB 环境而独立运行。MathWorks 公司希望使 MATLAB 成为新一代的通用软件开发工具，并为此提供了将 MATLAB 源程序编译为独立于 MATLAB 集成环境运行的 EXE 文件以及将 MATLAB 程序转化为 C 语言程序的编译器。

4. MATLAB 工具箱

MATLAB 包含两部分内容：基本部分和各种可选的工具箱。基本部分构成了 MATLAB 的核心内容，也是使用和构造工具箱的基础。MATLAB 工具箱分为两大类：功能性工具箱和学科性工具箱。功能性工具箱主要用来扩充其符号计算功能、可视建模仿真功能及文字处理功能等。学科性工具箱专业性比较强，如控制系统工具箱(Control System Toolbox)、信号处理工具箱(Signal Processing Toolbox)、神经网络工具箱(Neural Network Toolbox)、最优化工具箱(Optimization Toolbox)、金融工具箱(Financial Toolbox)、统计学工具箱 (Statistics Toolbox)，等等，这些工具箱都是由该领域内学术水平很高的专家编写的，用户可以直接利用这些工具箱进行相关领域的科学研究。

MATLAB具备很强的开放性。除内部函数外,所有MATLAB基本文件和各工具箱文件都是可读可改的源文件,用户可通过对源文件的修改或加入自己编写的文件去构建新的专用工具箱。

1.3 MATLAB系统的运行环境与安装

1.3.1 运行环境

MATLAB 5.3作为Windows 95/98下的一个应用程序,本身对软硬件没有特殊要求。也就是说,它对环境的要求与Windows 95/98要求是一致的。

1. 硬件环境

一般要求486以上的处理器、16 MB以上内存、足够的硬盘可用空间(随安装MATLAB组件的多少而定)、CD-ROM驱动器、鼠标等。

2. 软件环境

Windows 95/98或以上版本的操作系统。当使用MATLAB笔记本时,需先安装Microsoft Word。如果用户想生成mex文件,需安装Microsoft C/C++、Borland C/C++或Watcom C/C++,或Microsoft Fortran PowerStation。如果用户想阅读完整的MATLAB的帮助信息,需预先安装Netscape Navigator 2.0以上版本或者是Microsoft Internet Explorer 3.0以上版本来阅读其中的超文本帮助信息,还需安装Adobe Acrobat Reader来阅读其中的PDF格式的帮助文件。

1.3.2 安装

MATLAB 5.3必须在Windows 95/98环境下用系统自带的安装程序setup.exe进行安装。步骤如下:

(1) 启动Windows 95/98。

(2) 将MATLAB 5.3光盘放入光驱,一般情况下,安装程序会自动运行。假如没有自动运行,则在"我的电脑"或"资源管理器"中双击setup.exe文件即可。运行setup.exe文件后,进入MATLAB安装过程,稍等片刻,将会出现Welcome to the MATLAB Setup的欢迎界面。

(3) 单击Next按钮,进入用户安装协议界面。

(4) 单击Yes按钮,进入用户信息注册界面。此时,需要在Name栏中填入自己的名字或代号;在Company栏中填入自己的工作单位名称;在Personal License Password栏中正确填入用户使用许可号(可以在有关文本文件中或光盘盒上查找并妥善保存)。只有将这些信息都正确输入以后,安装过程才能继续。

(5) 单击Next按钮进入下一步。在Components列表框中,根据自身需要选择安装所需要的MATLAB组件。第一项为MATLAB基本组件,必须安装;其余的为MATLAB工具箱,可以根据需要进行选择。在缺省状态下,MATLAB 5.3所安装的路径为c:\matlabr11。本书假定选择缺省安装路径。如需修改,单击Browse按钮,进入路径选择界面,否则跳过这一步。注意,硬盘可用空间必须比MATLAB所需空间大。

(6) 单击 Next 按钮,进入下一步,进行文件复制。

(7) 文件复制结束,单击 Finish 按钮,MATLAB 安装完毕。

1.4 MATLAB 系统的启动与退出

1.4.1 启动

与一般的 Windows 95/98 程序一样,启动 MATLAB 系统有 3 种常见方法:

(1) 在 Windows 95/98 桌面,单击任务栏上的"开始"按钮,选择"程序"菜单项,然后单击 Matlab 菜单项中的 MATLAB 5.3 程序,就可启动 MATLAB 系统,这时将看到图 1.1 所示的 MATLAB 命令窗口。

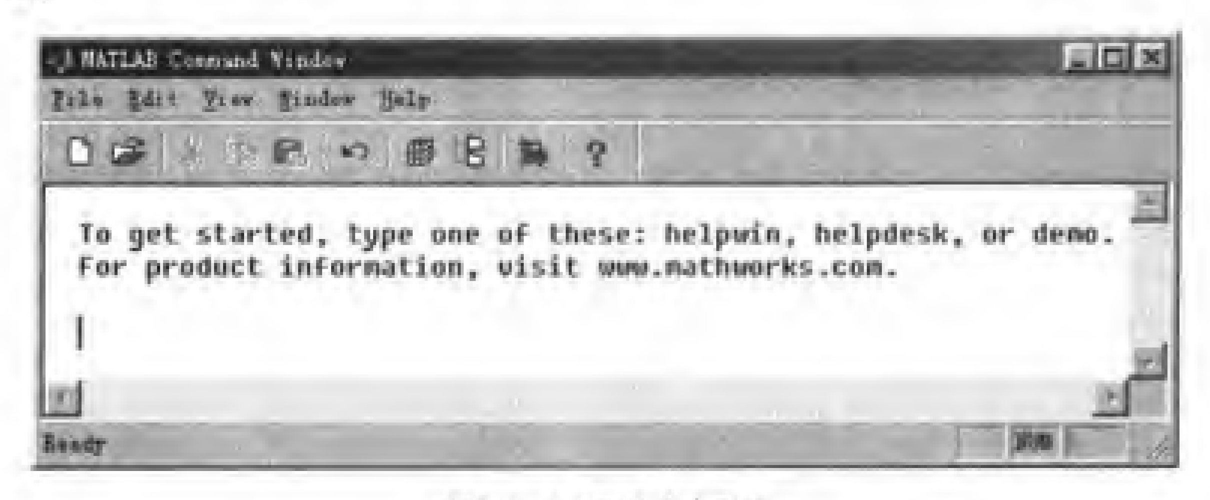


图 1.1 MATLAB 命令窗口

(2) 运行 c:\matlabr11\bin 中的 MATLAB 系统启动程序 matlab.exe。通过"我的电脑"或"资源管理器"去查找这个程序,然后双击它。

(3) 利用建立快捷方式的功能,将 MATLAB 系统启动程序以快捷方式放在 Windows 95/98 桌面上,只要在桌面上双击该图标即可启动 MATLAB。

1.4.2 退出

要退出 MATLAB 系统,也有 3 种常见方法:

(1) 在 MATLAB 命令窗口 File 菜单中选择 Exit MATLAB 命令。

(2) 在 MATLAB 命令窗口输入 Exit 或 Quit 命令。

(3) 单击 MATLAB 命令窗口的关闭按钮。

1.5 MATLAB命令窗口

启动MATLAB 5.3后，屏幕上会出现MATLAB命令窗口（如图1.1所示），它提供了用户和MATLAB交互操作的环境。MATLAB命令窗口的组成和一般Windows 95/98窗口的组成类似。在图1.1中，最上面显示MATLAB Command Window字样的一栏为标题栏，标题栏左边为窗口控制按钮，右边依次为窗口最小化按钮、窗口缩放按钮和关闭窗口按钮。标题栏下面为菜单栏，其中包含有5个菜单项。菜单栏下面为工具栏，其中提供了10个工具按钮。工具栏下面是命令编辑区，命令编辑区占窗口的绝大部分。

1.5.1 菜单栏

在MATLAB 5.3命令窗口的菜单栏，共包含File、Edit、View、Window和Help等5个菜单项。下面介绍它们的作用。

1. File菜单项

File菜单项实现有关文件的操作，所包含的命令如图1.2所示。



图1.2 File菜单项

File菜单项中各命令的作用如下：

(1) New命令　用于建立M文件、图形窗口或Simulink的模型。

(2) Open命令　打开一个已经建立的M文件。

(3) Open Selection 命令　打开选中的 M 文件。

(4) Run Script 命令　执行一个命令文件。

(5) Load Workspace 命令　将 MAT 文件中的变量装入到 MATLAB 工作空间。

(6) Save Workspace As 命令　把 MATLAB 工作空间的所有变量保存为 MAT 文件。

(7) Show Workspace 命令　打开工作空间浏览器,其中显示 MATLAB 工作空间中所有变量的类型、大小及占用的存储空间。

(8) Show Graphics Property Editor 命令　打开图形属性编辑器。在图形属性编辑器里,可以选择 MATLAB 已经打开的图形窗口,对每个图形对象的属性值进行修改。这部分内容将在第 7 章详细介绍。

(9) Show GUI Layout Tool 命令　打开图形界面控制面板。通过图形界面控制面板可以设计图形用户界面。这部分内容将在第 7 章详细介绍。

(10) Set Path 命令　打开路径浏览器。通过路径浏览器可以更改 MATLAB 执行命令时的搜索路径。

(11) Preferences 命令　打开命令窗口的环境设置卡,用于设置命令窗口所采用的显示格式、字体和图形复制选项。

(12) Print Setup 命令　设置打印机的参数,如打印机的类型、纸张大小、送纸方向、图形的打印质量,等等。

(13) Print 命令　用于打印命令窗口中的内容,也可以设置一些打印参数。

(14) Print Selection 命令　按照 Print Setup 选项的设置,打印选中的内容。

(15) Exit MATLAB 命令　退出 MATLAB 系统。

2. Edit 菜单项

Edit 菜单项用于命令窗口的编辑操作,如图 1.3 所示。

Edit 菜单项中各命令的作用如下:

(1) Undo、Cut、Copy 和 Paste 命令　分别用于撤销上一次操作、剪切、复制和粘贴,这 4 项和其他 Windows 95/98 应用程序的相应菜单项没有什么差别。

(2) Clear 命令　在命令编辑区输入命令出现错误时,选中想删除的内容,再单击 Clear 选项,将清除错误的内容,但已经按回车键的命令不可清除。



图 1.3　Edit 菜单项

(3) Select All 命令　用于选定文本编辑区的所有内容,以便进一步复制。

(4) Clear Session 命令　清除命令编辑区的全部内容,但并不删除工作空间中的变量。

3. View 菜单项

MATLAB 5.3 版比早期版本多提供了一个 View 菜单项,该菜单可以用来显示和关闭工具栏。

4. Window 菜单项

利用 Window 菜单项可以查看目前 MATLAB 打开的所有窗口,并可选中某个窗口为当前窗口,从而实现在不同窗口之间的转换。

5. Help菜单项

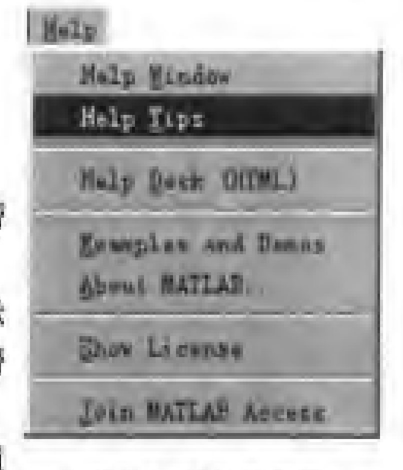


图1.4 Help菜单项

Help菜单项提供帮助信息，如图1.4所示。

Help菜单项中各命令的作用如下：

(1) Help Window命令　打开MATLAB的帮助窗口。

(2) Help Tips命令　打开帮助窗口，并首先显示MATLAB的帮助系统的分类和使用方法。

(3) Help Desk(HTML)命令　打开系统WWW浏览器，并显示MATLAB的帮助桌面。通过帮助桌面可以得到MATLAB的详细帮助。

(4) Examples and Demos命令　MATLAB演示窗主页。可以通过演示MATLAB提供的例子来熟悉相关部分的用法。

(5) About MATLAB命令　显示关于MATLAB的版本、版权等信息。

(6) Show License命令　显示软件许可协议。

(7) Join MATLAB Access命令　打开系统WWW浏览器，上网用户可通过填写相关的表格来获得MathWorks公司的产品。

1.5.2 工具栏

MATLAB 5.3命令窗口的工具栏共提供了10个命令按钮，按钮名称及功能如图1.5所示。这些命令按钮均有对应的菜单命令，但比菜单命令使用起来更快捷、方便。



图1.5 命令窗口的工具栏

1.5.3 命令编辑区

1. 命令编辑区的作用

命令编辑区用于输入命令和显示计算结果。选择File菜单项的Preference命令，可以设置命令编辑区的显示风格。有时在命令编辑区会出现命令提示符》，》表示MATLAB已准备好，正等待用户输入命令。在》提示符后键入命令并按下回车键后，MATLAB就会解释执行所输入的命令，并在命令后面给出计算结果。值得注意的是，在中文Windows环境下，不会出现提示符》，但这并不影响MATLAB的使用。

2. 命令行的输入规则

一般来说，一个命令行输入一条命令，命令行以回车结束。但一个命令行也可以输入若干条

命令，各命令之间以逗号分隔，若前一命令后带有分号，则逗号可以省略。例如

```
p = 15,m = 35
p = 15;m = 35
```

以上两个命令行都是合法的，第一个命令行执行后显示 p 和 m 的值，第二个命令行因命令 p = 15 后面带有分号，p 的值不显示，而只显示 m 的值。

如果一个命令行很长，一个物理行之内写不下，可以在第一个物理行之后加上 3 个小黑点并按下回车键，然后接着下一个物理行继续写命令的其他部分。3 个小黑点称为续行符，即把下面的物理行看作该行的“逻辑”继续。例如

```
s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 - ...
   1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

这是一个命令行，但占用 2 个物理行，第一个物理行以续行符结束，第二个物理行是上一行的继续。

3. 命令行的编辑

在 MATLAB 里，有很多的控制键和方向键可用于命令行的编辑。如果能熟练使用这些键将大大提高操作效率。例如，当将命令 x1 = (log(3) + sqrt(5))/2 中的函数名 sqrt 输入成 srt 时，由于 MATLAB 中不存在 srt 函数，MATLAB 将会给出错误信息：

```
??? Undefined function or variable ' srt '.
```

重新输入命令时，用户不用输入整行命令，而只需按 ↑ 键调出刚才输入的命令行，再在相应的位置输入 q 字母并按下回车键即可。在回车时，光标可以在该命令行的任何位置，没有必要将光标移到该命令行的末尾。反复使用 ↑ 键，可以回调以前输入的所有命令行，其作用类似 MS - DOS 下的 DOSKEY 命令。还可以只输入少量的几个字母，再按 ↑ 键就可以调出最后一条以这些字母开头的命令。例如，输入 plo 后再按 ↑ 键，则会调出最后一次使用的以 plo 开头的命令行。表 1.1 介绍了 MATLAB 命令行编辑的常用控制键及其功能。

表 1.1 命令行编辑的常用控制键

键 名	功 能	键 名	功 能
↑	前寻式调回已输入过的命令	Home	将光标移到当前行首端
↓	后寻式调回已输入过的命令	End	将光标移到当前行末尾
←	在当前行中左移光标	Del	删除光标右边的字符
→	在当前行中右移光标	Backspace	删除光标左边的字符
PgUp	前寻式翻滚一页	Esc	删除当前行全部内容
PgDn	后寻式翻滚一页		

4. 常用操作系统命令

除了菜单操作之外，MATLAB 还提供了许多直接在命令窗口执行的操作命令，以实施有关操作。下面先介绍类似于操作系统命令的常用命令（见表 1.2），其作用与 DOS 下相关命令类似。其他命令将在今后介绍相关内容时进行介绍。

表1.2　常用操作系统命令

命　令	功　能	命　令	功　能
dir	显示文件目录清单	more	使其后的显示内容分屏进行
cd	改变当前工作目录	copyfile	复制文件
mkdir	建立目录	what	显示当前目录下的M文件和MAT文件
delete	删除文件	clc	清除命令窗口显示的内容
type	显示文本文件内容	web	打开网络浏览器

表1.2中所列命令是在MATLAB命令窗口执行的命令，虽然和DOS的有关命令作用类似，但命令名、格式、用法并非完全相同，使用时不要混淆。

1.6　MATLAB文件管理

1.6.1　MATLAB的目录结构

为了对文件进行有效的组织和管理，MATLAB有自己严谨的目录结构，不同类型的文件放在不同的目录下。前面介绍MATLAB的安装时，选择了缺省安装路径c:\matlabr11，在安装目录下有下面几个常用的子目录。

(1) bin子目录　存放MATLAB的大部分可执行文件，库文件以及一些设置路径的批处理文件。

(2) extern子目录　存放MATLAB语言和C的接口，如头文件等。

(3) help子目录　存放HTML型的或PDF型的帮助文件。

(4) notebook子目录　MATLAB笔记本目录，即存放MATLAB语言和Word的接口。

(5) simulink子目录　Simulink软件所在的目录。

(6) toolbox子目录　各个工具箱所在目录。其中，matlab子目录下为MATLAB系统的基本内容，而其他的目录随所安装工具箱的不同而有所差异。

(7) work子目录　用户工作目录。

1.6.2　MATLAB的搜索路径

当用户在MATLAB命令窗口输入一条命令后，MATLAB按照一定次序寻找相关的文件。基本的搜索过程是：

(1) 检查是不是一个变量。

(2) 检查是不是一个内部函数。

(3) 检查是否当前目录下的M文件(扩展名为.m的文件)。

(4) 检查是否MATLAB搜索路径中其他目录下的M文件。

假定建立了一个变量examp，同时在当前目录下建立了一个M文件examp.m，如果在命令窗

口输入 examp,按照上面介绍的搜索过程,应是在屏幕上显示变量 examp 的值。如果没有建立 examp 变量,则执行 examp.m 文件。

1.6.3 用户目录的设置

用户可以将自己的工作目录设置成当前目录,从而使得用户的操作都在用户目录中进行。将用户目录设置成当前目录使用 cd 命令。例如,将用户目录 c:\mydir 设置为当前目录,可在命令窗口输入命令:

```
cd  c:\mydir
```

注意:设置的当前目录只是在当前启动的 MATLAB 环境下有效,一旦 MATLAB 重新启动,必须重新设置。

1.6.4 搜索路径的设置

用户可以将自己的工作目录列入 MATLAB 搜索路径,从而将用户目录纳入 MATLAB 系统统一管理。

1. 用 path 命令设置搜索路径

使用 path 命令可以把用户目录临时纳入搜索路径。例如,将用户目录 c:\mydir 加到搜索路径下,可在命令窗口输入命令:

```
path(path,'c:\mydir')
```

要注意的是,设置的搜索路径仅在当前启动的 MATLAB 环境下有效,一旦 MATLAB 重新启动,必须重新设置。

2. 用路径浏览器设置搜索路径

MATLAB 路径浏览器可用来设置当前目录和永久性改变搜索路径。

(1) 路径浏览器的启动

启动路径浏览器有 3 种方法:在 MATLAB 命令窗口 File 菜单中选 Set Path 命令;在命令窗口执行 pathtool 命令;单击命令窗口工具栏上的路径浏览器按钮。

启动路径浏览器后,屏幕上出现图 1.6 所示的路径浏览器窗口。

(2) 路径浏览器的操作

在路径浏览器窗口的 Path 菜单中选 Add to Path 命令,可将指定目录添加到搜索路径下。选择该命令后,将打开一个图 1.7 所示的添加目录对话框,用户可以在此对话框空白栏直接输入所需添加的目录,或通过空白栏右边的按钮选择所需添加的目录。再通过空白栏下方的单选按钮,决定把用户目录放在搜索路径前端还是后端。选择结束后,单击 OK 按钮,回到 MATLAB 路径浏览器。在路径浏览器中,选择 File 菜单项中的 Save Path 命令,便完成了添加操作。

在路径浏览器窗口的 Path 菜单中选 Remove from Path 命令,可将指定目录从搜索路径中删除。先在路径浏览器的路径列表中选定某目录,再选择 Path 中的 Remove from Path 命令,并选择 File 中的 Save Path 命令,即完成操作。

也可利用路径浏览器进行当前目录设置。在路径浏览器窗口单击 Browse 按钮,然后在出现的目录中选定所需的目录,便完成了设置。要注意的是,当前目录的设置都是临时的,一旦 MATLAB 重新启动,必须重新设置。

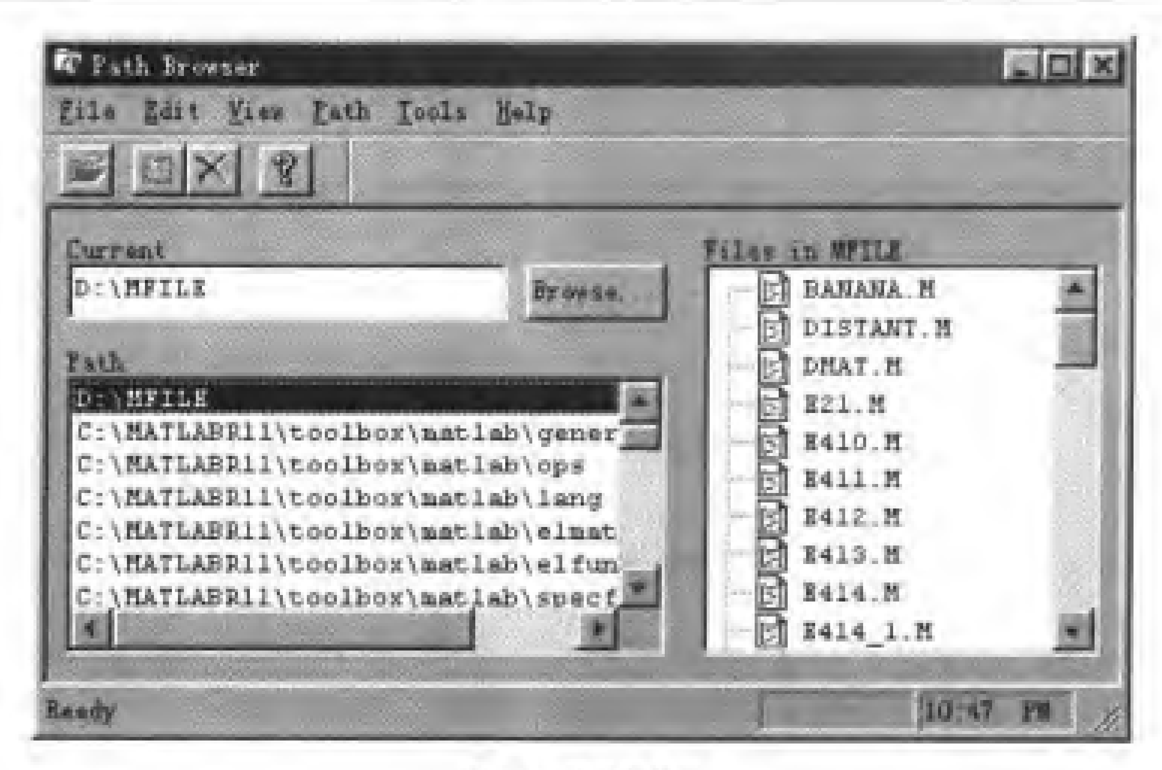


图1.6 路径浏览器



图1.7 添加目录对话框

1.7 MATLAB帮助系统

MATLAB提供了数目繁多的函数和命令，要全部把它们记下来是不现实的。可行的办法是先掌握一些基本内容，然后在实践中不断总结和积累，掌握其他内容。通过软件系统本身提供的帮助功能来学习软件的使用是重要的学习方法。MATLAB也提供了丰富的帮助功能，通过这种功能可以很方便地获得有关函数和命令的使用方法。

在MATLAB环境下，要获得帮助可通过以下几种方法：帮助命令、帮助窗口、帮助桌面或在线帮助页。对于Internet用户，还可直接链接到MathWorks公司的网页上(http://www.mathworks.com)寻求帮助。

1.7.1 帮助命令

1. help 命令

help 命令是查询函数语法的最基本方法，查询信息直接显示在命令窗口。例如，为了显示 sqrt 函数的使用方法与功能，可使用命令：

```
help sqrt
```

屏幕显示帮助信息：

```
SQRT Square root.
  SQRT(X) is the square root of the elements of X. Complex
  results are produced if X is not positive.
  See also SQRTM.
```

值得注意的是，MATLAB 命令窗口里显示的帮助信息用大写来突出函数名，但在使用函数时，要用小写。

MATLAB 按照函数的不同用途分别存放在不同的子目录下，用相应的帮助命令可显示某一类函数。例如，所有的线性代数函数均收在 matfun 子目录下，用命令

```
help matfun
```

可显示所有线性代数函数。

不带任何参数的 help 命令将显示所有的子目录。

2. lookfor 命令

help 命令只搜索出与那些关键字完全匹配的结果，lookfor 命令对搜索范围内的 M 文件进行关键字搜索，条件比较宽松。例如，因为不存在 inverse 函数，命令

```
help inverse
```

搜索结果为：

```
inverse.m not found.
```

而执行命令：

```
lookfor inverse
```

将得到 M 文件中包含 inverse 的全部函数。

lookfor 命令只对 M 文件的第一行进行关键字搜索。若在 lookfor 命令后加上-all 选项，则可对 M 文件进行全文搜索。

1.7.2 帮助窗口

帮助窗口给出的帮助信息和帮助命令给出的帮助信息一样，但在帮助窗口更容易浏览与之相关的其他函数。在 MATLAB 命令窗口中，要进入帮助窗口有 3 种方法：

(1) 单击工具栏上的帮助按钮。

(2) 键入 helpwin 命令。

(3) 选择 Help 菜单中的 Help Window 命令。

1.7.3 帮助桌面

帮助桌面比帮助命令及帮助窗口提供的帮助信息更多。帮助桌面的帮助信息大都用超文本

标记语言(HTML)写成,可以通过 Netscape 或 Internet Explorer 阅读。在 MATLAB 的命令窗口中,进入帮助桌面有两种方法:

(1) 键入 helpdesk 命令。

(2) 选择 Help 菜单的 Help Desk 命令。

这两种方法都会自动启动浏览器,用户可利用浏览器来浏览帮助信息。如果知道想要查询的函数的名字,可以使用 doc 命令直接查看帮助信息。doc 会自动打开浏览器并定位到相应函数。

1.7.4 在线帮助页

帮助桌面的所有文件均有相应的 PDF 格式文件,称为在线帮助页。PDF 格式文件可用 Adobe Acrobat Reader 阅读。用户选中帮助桌面上关于 PDF 格式文件的选项,或是在命令窗口中键入命令 doc,便会自动打开 Adobe Acrobat Reader。

对于 Internet 用户,还可以通过帮助桌面很方便地访问 MathWorks 公司的主页,向 MathWorks 公司咨询。

1.8 MATLAB 功能演示

本节通过几个有代表性的例子来演示 MATLAB 的功能,目的是使读者能初步领略到 MATLAB 的风格与特点。作为操作练习,读者可在 MATLAB 软件环境下验证下面的例子。

例 1.1 求解线性方程组:$\begin{cases} 2x+3y-z=2 \\ 8x+2y+3z=4 \\ 45x+3y+9z=23 \end{cases}$

在 MATLAB 命令窗口输入命令:

```
a=[2,3,-1;8,2,3;45,3,9];
b=[2;4;23];
x=inv(a)*b
```

其中前两条命令建立系数矩阵 $\boldsymbol{a}$ 和列向量 $\boldsymbol{b}$,第三条命令求根。inv(a)为 $\boldsymbol{a}$ 的逆矩阵,也可用 x=a\b 求根。得到的结果为:

```
x =
      0.5531
      0.2051
     -0.2784
```

例 1.2 绘制正弦曲线和余弦曲线。

在 MATLAB 命令窗口输入:

```
x=[0:0.5:360]*pi/180;
plot(x,sin(x),x,cos(x));
```

其中,第一条命令建立 $\boldsymbol{x}$ 向量,$\boldsymbol{x}$ 从 0°变化到 360°并转换为弧度,第二条命令绘制曲线。命令执行后,将打开一个图形窗口,并在其中显示正弦曲线和余弦曲线。

例 1.3 输入10个学生的成绩并对成绩按升序排序。

在 MATLAB 命令窗口输入：

```
g = input('请输入学生成绩：');
g = sort(g)
```

第一条命令执行后，屏幕提示“请输入学生成绩：”，在提示信息后输入 10 个学生的成绩，第二条命令将成绩按升序排序。假定输入 10 个学生的成绩为：

```
[56,78,67,89,78,54,65,90,89,97]
```

则输出结果为：

```
g =
   54   56   65   67   78   78   89   89   90   97
```

例 1.4 设有常微分方程初值问题：

$$y' = \frac{y^2 - t - 2}{4(t+1)}, 0 \leqslant t \leqslant 1$$

$$y(0) = 2$$

试求其数值解，并与精确解相比较(精确解为 $y(t) = \sqrt{t+1} + 1$)。

首先建立函数文件 funt.m：

```
function yp = funt(t,y)
yp = (y^2 - t - 2)/4/(t + 1);
```

然后求解微分方程：

```
t0 = 0;tf = 10;y0 = 2;
[t,y] = ode23('funt',[t0,tf],y0);
y1 = sqrt(t + 1) + 1;
t'
ans =
  Columns 1 through 7
         0    0.3200    0.9380    1.8105    2.8105    3.8105    4.8105
  Columns 8 through 13
    5.8105    6.8105    7.8105    8.8105    9.8105   10.0000
y'
ans =
  Columns 1 through 7
    2.0000    2.1490    2.3929    2.6786    2.9558    3.1988    3.4181
  Columns 8 through 13
    3.6198    3.8079    3.9849    4.1529    4.3133    4.3430
y1'
ans =
  Columns 1 through 7
    2.0000    2.1489    2.3921    2.6765    2.9521    3.1933    3.4105
  Columns 8 through 13
    3.6097    3.7947    3.9683    4.1322    4.2879    4.3166
```

其中 y 为数值解，$y1$ 为精确值，显然两者近似。

习 题 一

1. 与其他高级语言相比,MATLAB 有哪些显著特点?

2. 怎样理解 MATLAB 的开放性? 试结合自己的专业领域,为 MATLAB 设计一个工具箱。例如,假定一个桥梁专家为 MATLAB 设计一个桥梁设计工具箱。

3. 当正常安装好 MATLAB 后,误把 Windows 95 子目录删除。当重新安装 Windows 95 后,是否要再安装 MATLAB?

4. 试将 MATLAB 系统的用户界面、操作方式与其他 Windows 95 应用程序(如 Word)作一对比,有哪些共同规律?

5. 先建立自己的工作目录,再分别用 path 命令和路径浏览器将自己的工作目录设置到 MATLAB 搜索路径下。用 help 命令能查询到自己的工作目录吗?

6. 查看 MATLAB 的目录结构,并检查自己的机器上安装了哪些工具箱。

7. 利用 MATLAB 的帮助功能分别查询 inv 函数、plot 函数以及 MATLAB 操作系统命令的功能及用法。

8. roots 函数可以用于求多项式的根。先用 help 命令查看该函数的用法,然后利用该函数求 $x^4+7x^3+9x-20=0$ 的全部根。

第2章　MATLAB 数据

随着 MATLAB 版本的提高,MATLAB 数据类型更为丰富,除整型、双精度型、字符型等基本数据类型外,还有结构体、单元等更为复杂的数据类型。而且,以上各种数据类型都以矩阵(在其他高级语言中也称数组)形式存在,所以矩阵是 MATLAB 最基本的数据对象。十分丰富的数据类型,使得 MATLAB 的数据表达能力非常强,给应用带来极大方便,也为 MATLAB 管理更复杂的数据提供可能,从而进一步扩展了 MATLAB 的应用领域。

本章首先分析 MATLAB 数据的特点,然后重点介绍 MATLAB 中各种数据的表示方法以及数据的各种基本运算,从而为学习 MATLAB 程序设计作好准备。

2.1　MATLAB 数据的特点

正如 MATLAB 的名字——"矩阵实验室"的含义一样,MATLAB 是由早期专门用于矩阵运算的软件发展而来。矩阵是 MATLAB 最基本、最重要的数据对象,MATLAB 的大部分运算或命令都是在矩阵运算的意义下执行的,而且这种运算定义在复数域上。正因为如此,MATLAB 的矩阵运算功能非常丰富,体现出比其他高级语言要高得多的编程效率。许多含有矩阵运算的复杂计算问题,在 MATLAB 中很容易得到解决。

因为向量可以看成是仅有一行或一列的矩阵,单个数据(标量)可以看成是仅含一个元素的矩阵,故向量和单个数据都可以作为特殊矩阵来处理。

在一般情况下,矩阵的每个元素必须具有相同的数据类型。对于数值数据,MATLAB 中最常用的类型为双精度型,占 64 位(8 个字节),用 double 函数实现转换。此外,还有单精度数,占 32 位(4 个字节),用 single 函数实现转换。还有带符号整数和无符号整数,其转换函数有 int8、int16、int32、uint8、uint16、uint32,每一函数名后面的数字表示相应数据类型所占位数,其含义不难理解。

除数值数据以外,还有字符数据,在 MATLAB 中用 char 函数实现转换。

在实际应用中,有时需要将不同类型的数据构成矩阵的元素,为此 MATLAB 提供了结构(Structure)和单元(Cell)数据类型。此外,还有多维矩阵以及工程中应用十分广泛的稀疏矩阵(Sparse)。

2.2　变量和赋值

2.2.1　变量的命名

变量代表一个或若干个内存单元,为了对变量所对应的存储单元进行访问,需要给变量命

名。在 MATLAB 中,变量名是以字母开头,后接字母、数字或下划线的字符序列,最多 19 个字符。例如,myexampl2、my_exampl2、myexampl2_均为合法的变量名,而 12myexamp、_myexampl2 为非法的变量名。另外,在 MATLAB 中,变量名区分字母的大小写。这样,myexamp、MYexamp 和 MYEXAMP 表示 3 个不同的变量。值得注意的是,MATLAB 提供的标准函数名以及命令名必须用小写字母。例如,求矩阵 A 的逆用 inv(A),不能写成 Inv(A)或 INV(A),否则会出错。

2.2.2 赋值语句

MATLAB 赋值语句有两种格式:

(1) 变量 = 表达式

(2) 表达式

其中表达式是用运算符将有关运算量连接起来的式子,其结果是一个矩阵。在第一种语句形式下,MATLAB 将右边表达式的值赋给左边的变量,而在第二种语句形式下,将表达式的值赋给 MATLAB 的预定义变量 ans。

一般地,运算结果在命令窗口中显示出来。如果在语句的最后加分号,那么,MATLAB 仅仅执行赋值操作,不再显示运算的结果。如果运算的结果是一个很大的矩阵或根本不需要运算结果,则可以在语句的最后加上分号。

在 MATLAB 语句后面可以加上注释,用于解释或说明语句的含义,对语句处理结果不产生任何影响。注释以%开头,后面是注释的内容。

例 2.1 计算表达式 $\frac{5+\cos 47^{\circ}}{1+\sqrt{7}-2i}$ 的值,并将结果赋给变量 x,然后显示出结果。

在 MATLAB 命令窗口输入命令:

```
x = (5 + cos(47 * pi/180))/(1 + sqrt(7) - 2 * i)          %计算表达式的值
```

其中 pi 和 i 都是 MATLAB 定义的变量,分别代表圆周率 π 和虚数单位。

输出结果是:

```
x =
   1.1980 + 0.6572i
```

2.2.3 数据的输出格式

MATLAB 用十进制数表示一个常数,具体可采用日常记数法和科学记数法两种表示方法。如 3.14159、-9.359i、3 + 5i 等是采用日常记数法表示的常数,与通常的数学表示一样。又如 1.78029e2、6.732E2i、1234e-3-5i 等是采用科学记数法表示的常数,在这里用字母 e 或 E 表示以 10 为底的指数。

在一般情况下,MATLAB 内部每一个数据元素都是用双精度数来表示和存储的。数据输出时用户可以用 format 命令设置或改变数据输出格式。format 命令的格式为:

```
format  格式符
```

其中格式符决定数据的输出格式,各种格式符及其含义见表 2.1。注意,format 命令只影响数据输出格式,而不影响数据的计算和存储。

表 2.1 控制数据输出格式的格式符

格式符	含 义
short	输出小数点后 4 位，最多不超过 7 位有效数字。对于大于 1 000 的实数，用 5 位有效数字的科学记数形式输出
long	15 位有效数字形式输出
short e	5 位有效数字的科学记数形式输出
long e	15 位有效数字的科学记数形式输出
short g	从 short 和 short e 中自动选择最佳输出方式
long g	从 long 和 long e 中自动选择最佳输出方式
rat	近似有理数表示
hex	十六进制表示
+	正数、负数、零分别用 +、-、空格表示
bank	银行格式，元、角、分表示
compact	输出变量之间没有空行
loose	输出变量之间有空行

如果输出矩阵的每个元素都是纯整数，MATLAB 就用不加小数点的纯整数格式显示结果。只要矩阵中有一个元素不是纯整数，MATLAB 将按当前的输出格式显示计算结果。缺省的输出格式是 short 格式。作为一个例子，假定输入为：

```
x = [4/3 1.2345e-6]
```

那么，在各种不同的格式符下的输出为：

短格式(short)：1.3333 0.0000

短格式 e 方式(short e)：1.3333e+00 1.2345e-06

长格式(long)：1.33333333333333 0.00000123450000

长格式 e 方式(long e)：1.33333333333333e+00 1.23450000000000e-06

银行格式(bank)：1.33 0.00

十六进制格式(hex)：3ff5555555555555 3eb4b6231abfd271

+格式(+)：+ +

2.2.4 预定义变量

在 MATLAB 工作空间中，还驻留几个由系统本身定义的变量。它们有特定的含义，在使用时，应尽量避免对这些变量重新赋值。除前面介绍过的 ans 外，还有一些常用的预定义变量，现将它们列于表 2.2 中。

表 2.2　常用的预定义变量

预定义变量	含　义	预定义变量	含　义
ans	计算结果的缺省赋值变量	nargin	函数输入参数个数
eps	机器零阈值	nargout	函数输出参数个数
pi	圆周率 π 的近似值	realmax	最大正实数
i,j	虚数单位	realmin	最小正实数
inf,Inf	无穷大。如 1/0 的结果	lasterr	存放最新的错误信息
NaN,nan	非数。如 0/0、inf/inf 的结果	lastwarn	存放最新的警告信息

2.2.5　内存变量的管理

1. 内存变量的显示与删除

who 和 whos 这两个命令用于显示在 MATLAB 工作空间中已经驻留的变量名清单。但 whos 在给出驻留变量名的同时，还给出它们的维数、所占字节数及性质。

clear 命令用于删除 MATLAB 工作空间中的变量。注意，预定义变量不能被删除。

2. 工作空间浏览器

(1) 工作空间浏览器的启动

MATLAB 工作空间浏览器专门用于内存变量的管理。启动工作空间浏览器有 3 种方法：在命令窗口 File 菜单中选 Show Workspace 命令；在命令窗口执行 workspace 命令；单击命令窗口工具栏上的工作空间浏览器按钮。

启动工作空间浏览器后，屏幕上出现图 2.1 所示的工作空间浏览器窗口。

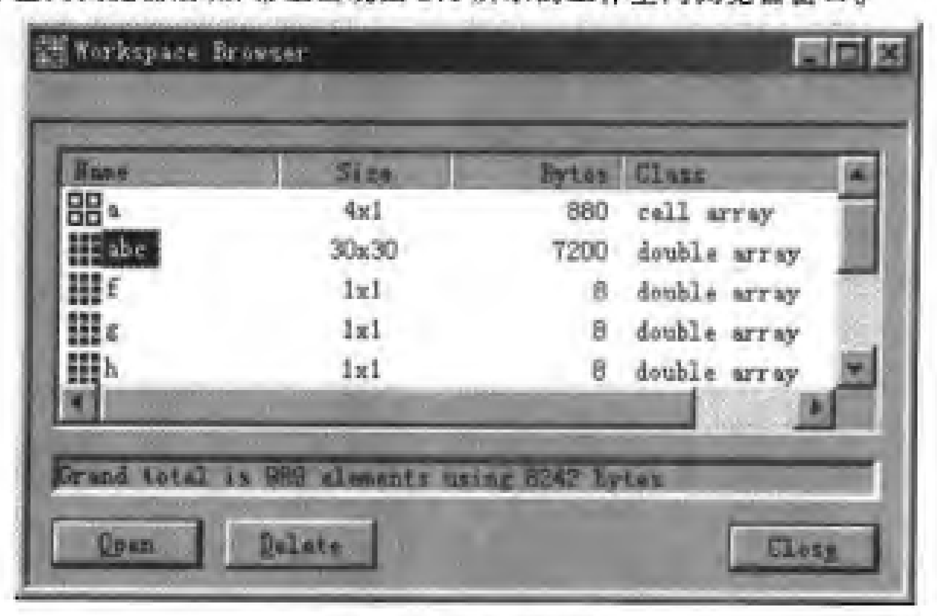


图 2.1　工作空间浏览器

(2) 工作空间浏览器的操作

工作空间浏览器显示所有内存变量的属性。当选中某些变量后，再单击 delete 按钮，这些变量将被删除。当选中某些变量后，再单击 open 按钮，将进入变量编辑器(图 2.2)。通过变量编辑器可以直接观察变量中的具体元素，也可修改变量中的具体元素。

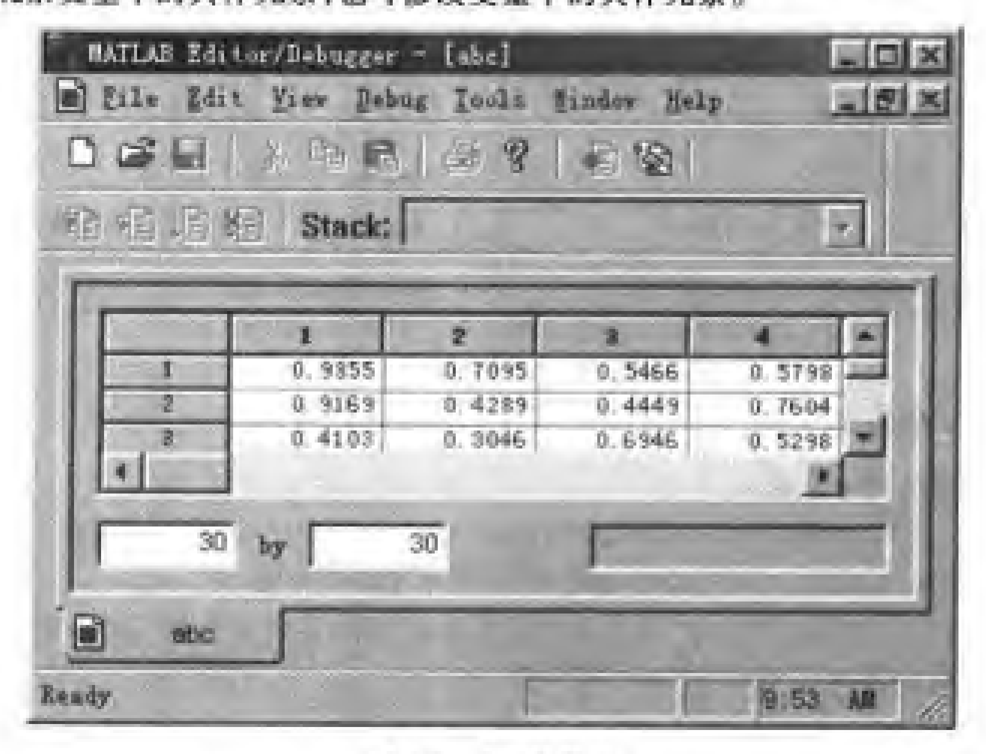


图 2.2 变量编辑器

通常，对于较大矩阵的输入，可采用变量编辑器。操作方法是：

① 在命令窗口中向一个新变量赋空阵。

② 在工作空间浏览器中打开该变量。

③ 在变量编辑器左下方的 2 个填充栏中填写待建矩阵的行数和列数，于是在变量编辑窗中就会出现空白表格。表格的每一个方格对应矩阵的一个元素，在方格中填写元素值。

3. 内存变量文件

利用 MAT 文件可以把当前 MATLAB 工作空间中的一些有用变量长久地保留下来。MAT 文件是 MATLAB 保存数据的一种标准格式二进制文件，扩展名一定是 .mat。MAT 文件的生成和装入由 save 和 load 命令来完成。常用格式为：

```
save 文件名 [变量名表] [-append][-ascii]
load 文件名 [变量名表] [-ascii]
```

其中，文件名可以带路径，但不需带扩展名 .mat，命令隐含一定对 .mat 文件进行操作。变量名表中的变量个数不限，只要内存或文件中存在即可，变量名之间以空格分隔。当变量名表省略时，保存或装入全部变量。-ascii 选项使文件以 ASCII 格式处理，省略该选项时文件将以二进制格式处理。save 命令中的 -append 选项控制将变量追加到 MAT 文件中。

假定变量 am 和 D 存在于 MATLAB 工作空间中，输入以下命令便可借助 mydata.mat 文件保存 am 和 D：

```
save mydata am D
```

假如在下次重新进入 MATLAB 后，需要使用矩阵 am 和 D，可用下述命令把 mydata.mat 中的内容装入 MATLAB 工作空间：

```
load mydata
```

在执行上述命令后，在当前的 MATLAB 环境中，am 和 D 就是两个已知变量了。

注意：mydata 是用户自已取的文件名，MATLAB 默认扩展名为.mat。上述 save 命令执行以后，该 mydata.mat 文件将存放在当前目录。假如用户有意要让 mydata.mat 存放在指定的其他目录(比如 d:\lpp 目录)上，那么 save 命令改为：

```
save d:\lpp\mydata am D
```

当然，相应 load 命令中文件名前也要加路径名。

除了操作命令以外，通过 MATLAB 命令窗口 File 菜单项中的 Save Workspace As 命令可以保存工作空间中的全部变量。相应地，通过 File 菜单项中的 Load Workspace 命令可以将保存在 MAT 文件中的变量装入到 MATLAB 工作空间。

2.3　MATLAB 矩阵

在 MATLAB 中，不需对矩阵的维数和类型进行说明，MATLAB 会根据用户所输入的内容自动进行配置。

2.3.1　矩阵的建立

1. 直接输入法

最简单的建立矩阵的方法是从键盘直接输入矩阵的元素。具体方法如下：将矩阵的元素用方括号括起来，按矩阵行的顺序输入各元素，同一行的各元素之间用空格或逗号分隔，不同行的元素之间用分号分隔。例如，键入命令：

```
A=[1 2 3;4 5 6;7 8 9]
A=
    1     2     3
    4     5     6
    7     8     9
```

这样，在 MATLAB 的工作空间中就建立了一个矩阵 ***A***，以后就可以使用矩阵 ***A***。

也可以用回车键代替分号，按下列方式输入：

```
A=[1  2  3
   4  5  6
   7  8  9]
```

MATLAB 提供对复数的操作与管理功能。在 MATLAB 中，虚数单位用 i 或 j 表示。例如，6+5i 与 6+5j 表示的是同一个复数。也可以写成 6+5*i 或 6+5*j，这里将 i 或 j 看作一个运算量参与表达式的运算。例如，建立复数矩阵：

```
a=exp(2);
B=[1,2+i*a,a*sqrt(a);sin(pi/4),a/5,3.5+6i]
```

```
B =
    1.0000     2.0000 + 7.3891i   20.0855
    0.7071     1.4778              3.5000  +  6.0000i
```

复数矩阵还可采用另一种输入方式。例如，

```
R = [1,2,3;4,5,6];
I = [6,7,8;9,10,11];
ri = R + i * I
ri =
    1.0000 + 6.0000i     2.0000 + 7.0000i      3.0000 + 8.0000i
    4.0000 + 9.0000i     5.0000 + 10.0000i     6.0000 + 11.0000i
```

在这里 i 是单个数据，i * I 表示一个数与一个矩阵相乘。

2. 利用 M 文件建立矩阵

对于比较大且比较复杂的矩阵，可以为它专门建立一个 M 文件。下面通过一个简单例子来说明如何利用 M 文件创建矩阵。

例 2.2　利用 M 文件建立 MYMAT 矩阵。

(1) 启动有关编辑程序或 MATLAB 文本编辑器(见第 3 章)，并输入待建矩阵：

```
MYMAT = [101,102,103,104,105,106,107,108,109;
         201,202,203,204,205,206,207,208,209;
         301,302,303,304,305,306,307,308,309];
```

(2) 把输入的内容以纯文本方式存盘(设文件名为 mymatrix.m)。

(3) 在 MATLAB 命令窗口中输入 mymatrix，即运行该 M 文件，就会自动建立一个名为 MYMAT 的矩阵，可供以后使用。

3. 利用 MATLAB 函数建立矩阵

MATLAB 提供了许多产生特殊矩阵的函数，可以利用它们去建立矩阵。下面先列举几个产生特殊矩阵的函数。

zeros	产生全 0 矩阵(零矩阵)
ones	产生全 1 矩阵(幺矩阵)
eye	产生单位矩阵
rand	产生 0 ~ 1 间均匀分布的随机矩阵
randn	产生 0 ~ 1 间正态分布的随机矩阵

这几个函数的调用格式相似，下面以产生零矩阵的 zeros 函数为例进行说明。其调用格式是：

zeros(m)	产生 $m \times m$ 零矩阵
zeros(m,n)	产生 $m \times n$ 零矩阵。当 $m = n$ 时，等同于 zeros(m)
zeros(size(A))	产生与矩阵 A 同样大小的零矩阵

其中，size(A)函数返回包含两个元素的向量，分别是矩阵 A 的行数和列数。相关的函数有：length(A)给出 A 的行数和列数中的较大者，即 length(A) = max(size(A))；ndims(A)给出 A 的维数。

例 2.3　分别建立 3×3、3×2 和与矩阵 A 同样大小的零矩阵。

(1) 建立一个 3×3 零矩阵：

```
zeros(3)
ans =
     0     0     0
     0     0     0
     0     0     0
```

(2) 建立一个 3×2 零矩阵：

```
zeros(3,2)
ans =
     0     0
     0     0
     0     0
```

(3) 设 ***A*** 为 2×3 矩阵，则可以用 zeros(size(A))建立一个与矩阵 ***A*** 同样大小零矩阵：

```
A = [1 2 3;4 5 6];        %产生一个 2×3 阶矩阵 A
zeros(size(A))            %一个与矩阵 A 同样大小的零矩阵
ans =
     0     0     0
     0     0     0
```

此外，常用的函数还有 reshape(A,m,n)，它在矩阵总元素保持不变的前提下，将矩阵 ***A*** 重新排成 m×n 的二维矩阵。例如

```
xv = [23,45,65,34,65,34,98,45,78,65,43,76];     %产生有 12 个元素的行向量 xv
ym = reshape(xv,3,4)                            %利用向量 xv 建立 3×4 矩阵 ym
ym =
    23    34    98    65
    45    65    45    43
    65    34    78    76
```

注意，在 MATLAB 中，矩阵元素按列存储，即首先存储矩阵的第一列元素，然后存储第二列元素，…，一直到矩阵的最后一列元素。reshape 函数只是改变原矩阵的行数和列数，即改变其逻辑结构，但并不改变原矩阵元素个数及它的存储结构。例如，再针对上面建立的矩阵 ***ym***，执行命令：

```
newym = reshape(ym,2,6)
```

则输出为：

```
newym =
    23    65    65    98    78    43
    45    34    34    45    65    76
```

4. 建立大矩阵

大矩阵可由方括号中的小矩阵建立起来。例如

```
A = [1 2 3;4 5 6;7 8 9];
C = [A,eye(size(A)); ones(size(A)),A]
C =
     1     2     3     1     0     0
     4     5     6     0     1     0
     7     8     9     0     0     1
```

```
1   1   1   1   2   3
1   1   1   4   5   6
1   1   1   7   8   9
```

2.3.2 冒号表达式

在 MATLAB 中，冒号是一个重要的运算符。利用它可以产生行向量。冒号表达式的一般格式是：

```
e1:e2:e3
```

其中 *e*1 为初始值，*e*2 为步长，*e*3 为终止值。冒号表达式可产生一个由 *e*1 开始到 *e*3 结束，以步长 *e*2 自增的行向量。例如

```
t=0:1:5
```

将产生行向量 *t*，各元素为 0,1,2,3,4,5。

在冒号表达式中如果省略 *e*2 不写，则步长为 1。例如，t=0:5 与 t=0:1:5 等价。

在 MATLAB 中，还可以用 linspace 函数产生行向量。其调用格式为：

```
linspace(a,b,n)
```

其中 *a* 和 *b* 是生成向量的第一个和最后一个元素，*n* 是元素总数。当 *n* 省略时，自动产生 100 个元素。显然，linspace(a,b,n)与 a:(b-a)/(n-1):b 等价。

2.3.3 矩阵的拆分

1. 矩阵元素

MATLAB 允许用户对一个矩阵的单个元素进行赋值和操作。例如，如果想将矩阵 ***A*** 的第 3 行第 2 列的元素赋为 200，则可以通过下面语句来完成：

```
A(3,2)=200
```

这时将只改变该元素的值，而不影响其他元素的值。如果给出的行下标或列下标大于原来矩阵的行数和列数，则 MATLAB 将自动扩展原来的矩阵，并将扩展后未赋值的矩阵元素置为 0。例如

```
A=[1,2,3;4,5,6];
A(4,5)=10
```

则输出为：

```
1   2   3   0   0
4   5   6   0   0
0   0   0   0   0
0   0   0   0   10
```

在 MATLAB 中，也可以采用矩阵元素的序号来引用矩阵元素。矩阵元素按列编号，先第一列，再第二列，依次类推。例如

```
A=[1,2,3;4,5,6];
A(3)
ans =
    2
```

显然,序号(Index)与下标(Subscript)是一一对应的,以 $m \times n$ 矩阵 A 为例,矩阵元素 $A(i,j)$ 的序号为 $(j-1)*m+i$。其相互转换关系也可利用 sub2ind 和 ind2sub 函数求得。

2. 矩阵拆分

(1) 利用冒号表达式获得子矩阵

① A(:,j)表示取 A 矩阵的第 j 列全部元素;A(i,:)表示 A 矩阵第 i 行的全部元素;A(i,j)表示取 A 矩阵第 i 行、第 j 列的元素。

② A(i:i+m,:)表示取 A 矩阵第 $i \sim i+m$ 行的全部元素;A(:,k:k+m)表示取 A 矩阵第 $k \sim k+m$ 列的全部元素,A(i:i+m,k:k+m)表示取 A 矩阵第 $i \sim i+m$ 行内,并在第 $k \sim k+m$ 列中的所有元素。例如

```
A=[1,2,3,4,5;6,7,8,9,10;11,12,13,14,15;16,17,18,19,20];
A(2:3,4:5)
ans =
      9     10
     14     15
```

又如

```
A(2:3,1:2:5)
ans =
      6      8      10
     11     13      15
```

利用 MATLAB 的冒号运算,可以很容易地从给出的矩阵中获得子矩阵,这样处理的速度比后面将介绍的利用循环语句来赋值的方式快得多,所以在实际编程时应该尽量采用这种赋值方法。

此外,还可利用一般向量和 end 运算符等来表示矩阵下标,从而获得子矩阵。end 表示矩阵某一维的末尾元素下标。例如

```
A=[1,2,3,4,5;6,7,8,9,10;11,12,13,14,15;16,17,18,19,20];
A(end,:);                      %取 A 最后一行元素
A([1,4],3:end)                 %取 A 第 1,4 两行中第 3 列到最后一列的元素
ans =
      3      4      5
     18     19     20
```

(2) 利用空矩阵删除矩阵的元素

在 MATLAB 中,定义[ ]为空矩阵。给变量 X 赋空矩阵的语句为 X=[ ]。注意,X=[ ]与 clear X 不同,clear 是将 X 从工作空间中删除,而空矩阵则存在于工作空间,只是维数为 0。

将某些元素从矩阵中删除,采用将其置为空矩阵的方法就是一种有效的方法。例如

```
A=[1 2 3 4 5 6;7 8 9 10 11 12;13 14 15 16 17 18];
A(:,[2 4])=[]
```

其中第二条命令将删除 A 的第二列和第四列元素。输出为:

```
A =
      1      3      5      6
      7      9     11     12
     13     15     17     18
```

2.3.4 多维矩阵

除了二维矩阵之外,MATLAB 5.x 还能处理三维或多维矩阵。三维矩阵的逻辑结构可以看作由若干页二维矩阵所组成。通常,三维矩阵的第一维称为行,第二维称为列,第三维称为页。对三维矩阵而言,无论哪一"页"上的"行,列"二维矩阵,其大小是相同的,同样,无论哪一"行"上的"列,页"二维矩阵,其大小也是相同的,无论哪一"列"上的"行,页"二维矩阵,其大小也是相同的。三维以上矩阵的形象思维比较困难,下面主要介绍三维矩阵。

三维矩阵的建立以二维矩阵为基础,常用的方法有 4 种:

(1) 对二维矩阵进行扩充得到三维矩阵。例如

```
a=[12,34,56,78;3,4,5,6];          %建立一个二维矩阵
a(2,5,1)=100;                     %对二维矩阵进行扩充
a(2,5,2)=100;
a(2,5,3)=100;
a                                 %显示三维矩阵 a
a(:,:,1)=
    12    34    56    78     0
     3     4     5     6   100
a(:,:,2)=
     0     0     0     0     0
     0     0     0     0   100
a(:,:,3) =
     0     0     0     0     0
     0     0     0     0   100
```

(2) 若干个同样大小的二维矩阵进行组合得到三维矩阵。例如

```
clear        %删除全部内存变量
a=[12,34,56,78;3,4,5,6];
b(:,:,1)=a;
b(:,:,2)=a;
b(:,:,3)=a;
b
b(:,:,1) =
    12    34    56    78
     3     4     5     6
b(:,:,2) =
    12    34    56    78
     3     4     5     6
b(:,:,3) =
    12    34    56    78
     3     4     5     6
```

(3) 除产生单位矩阵的 eye 函数外,前面介绍的建立矩阵的函数都可以延伸到三维矩阵。例如

```
ar = rand(2,4,3)
ar(:,:,1) =
              0.6038    0.1988    0.7468    0.9318
              0.2722    0.0153    0.4451    0.4660
ar(:,:,2) =
              0.4186    0.5252    0.6721    0.0196
              0.8462    0.2026    0.8381    0.6813
ar(:,:,3) =
              0.3795    0.5028    0.4289    0.1897
              0.8318    0.7095    0.3046    0.1934
size(ar)
ans =
      2     4     3
ndims(ar)
ans =
      3
```

(4) 用 cat 函数构建多维矩阵。一般调用格式是：

```
cat(n,A1,A2,…,An)
```

cat 函数把大小相同的若干矩阵,沿第 n 维方向串接成高维矩阵。当 $n=1$ 和 2 时,沿行和列的方向串接,结果是二维矩阵。当 $n=3$ 时,沿页的方向串接,结果是三维矩阵。例如

```
a = [-1,-3,-2;3,4,5];
b = cat(1,a,a*(-1),a*2)      %沿行的方向串接
b =
     -1     -3     -2
      3      4      5
      1      3      2
     -3     -4     -5
     -2     -6     -4
      6      8     10
c = cat(2,a,a*(-1),a*2)      %沿列的方向串接
c =
-1     -3     -2      1      3      2     -2     -6     -4
 3      4      5     -3     -4     -5      6      8     10
d = cat(3,a,a*(-1),a*2)      %沿页的方向串接
d(:,:,1) =
    -1     -3     -2
     3      4      5
d(:,:,2) =
     1      3      2
    -3     -4     -5
d(:,:,3) =
```

```
   -2    -6    -4
    6     8    10
```

2.4 MATLAB 运算

2.4.1 算术运算

1. 基本算术运算

MATLAB 的基本算术运算有：+（加）、-（减）、*（乘）、/（右除）、\（左除）、^（乘方）。这些算术运算的运算规则不难理解，但必须注意，运算是在矩阵意义下进行的，单个数据的算术运算只是一种特例。

(1) 矩阵加减运算

假定有两个矩阵 **A** 和 **B**，则可以由 **A** + **B** 和 **A** - **B** 实现矩阵的加减运算。运算规则是：若 **A** 和 **B** 矩阵的维数相同，则可以执行矩阵的加减运算，**A** 和 **B** 矩阵的相应元素相加减。如果 **A** 与 **B** 的维数不相同，则 MATLAB 将给出错误信息，提示用户两个矩阵的维数不匹配。

一个标量也可以和其他不同维数的矩阵进行加减运算。例如

```
x = [2, -1,0;3,2, -4];
y = x - 1
y =
    1    -2    -1
    2     1    -5
```

(2) 矩阵乘法

假定有两个矩阵 **A** 和 **B**，若 **A** 为 $m \times n$ 矩阵，**B** 为 $n \times p$ 矩阵，则 **C** = **A** * **B** 为 $m \times p$ 矩阵，其各个元素为

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}, \quad i = 1,2,\cdots,m; j = 1,2,\cdots,p$$

例如

```
A = [1,2,3;4,5,6];B = [1,2;3,0;7,4];
C = A * B
C =
      28      14
      61      32
```

矩阵 **A** 和 **B** 进行乘法运算，要求 **A** 的列数与 **B** 的行数相等，此时则称 **A**，**B** 矩阵是可乘的，或称 **A** 和 **B** 两矩阵维数相容。如果两者的维数不相容，则将给出错误信息，通知用户两个矩阵是不可乘的。

在 MATLAB 中还可以进行矩阵和标量相乘，标量可以是乘数也可以是被乘数。矩阵和标量相乘就是矩阵中的每个元素与此标量相乘。

(3) 矩阵除法

在 MATLAB 中有两种矩阵除法运算：\ 和/，分别表示左除和右除。如果 **A** 矩阵是非奇异方阵，则 A \B 和 B/A 运算可以实现。A \B 等效于 **A** 的逆左乘 **B** 矩阵，也就是 inv(A) * B，而 B/A 等效于 **A** 矩阵的逆右乘 **B** 矩阵，也就是 B * inv(A)。

对于含有标量的运算，两种除法运算的结果相同，如 3/4 和 4 \3 有相同的值，都等于 0.75。又如，设 a = [10.5,25]，则 a/5 = 5 \a = [2.1000 5.0000]。对于矩阵来说，左除和右除表示两种不同的除数矩阵和被除数矩阵的关系。对于矩阵运算，一般 A \B≠B/A。例如

```
a = [1 2 3;4 2 6;7 4 9];
b = [4,3,2;7,5,1;12,7,92];
c1 = a \ b
c2 = b/a
```

输出结果分别是：

```
c1 =
      0.5000      -0.5000      44.5000
      1.0000       0.0000      46.0000
      0.5000       1.1667     -44.8333
c2 =
      -0.1667      -3.3333          2.5000
      -0.8333      -7.6667          5.5000
       12.8333      63.6667       -36.5000
```

(4) 矩阵的乘方

一个矩阵的乘方运算可以表示成 A^x，要求 **A** 为方阵，x 为标量。例如

```
A = [1,2,3;4,5,6;7,8,0];
A ^ 2
ans =
      30      36      15
      66      81      42
      39      54      69
```

显然，A^2 即 A * A。

矩阵的开方运算是相当困难的，但有了计算机，这种运算就不再显得那么麻烦了，用户可以利用计算机方便地求出一个矩阵的方根。例如

```
A = [1,2,3;4,5,6;7,8,0];
A^0.1
ans =
      0.9750 + 0.2452i    0.1254 - 0.0493i    0.0059 - 0.0604i
      0.2227 - 0.0965i    1.1276 + 0.1539i    0.0678 - 0.1249i
      0.0324 - 0.1423i    0.0811 - 0.1659i    1.1786 + 0.2500i
```

2. 点运算

在 MATLAB 中，有一种特殊的运算，因为其运算符是在有关算术运算符前面加点，所以叫点运算。点运算符有 .* 、./、.\ 和 .^。两矩阵进行点运算是指它们的对应元素进行相关运算，要求两矩阵的维参数相同。例如

```
A=[1,2,3;4,5,6;7,8,9];
B=[-1,0,1;1,-1,0;0,1,1];
C=A.*B
C=
    -1     0     3
     4    -5     0
     0     8     9
```

A.*B 表示 ***A*** 和 ***B*** 单个元素之间对应相乘。显然与 A*B 的结果不同。

如果 ***A***、***B*** 两矩阵具有相同的维数,则 A./B 表示 ***A*** 矩阵除以 ***B*** 矩阵的对应元素。B.\A 等价于 A./B。例如

```
x=[1 2 3];y=[4 5 6];
z1=x./y
z2=y.\x
```

输出分别是:

```
z1=
    0.2500    0.4000    0.5000
z2=
    0.2500    0.4000    0.5000
```

显然 x./y 和 y.\x 值相等。这与前面介绍的矩阵的左除、右除是不一样的。

若两个矩阵的维数一致,则 A.^B 表示两矩阵对应元素进行乘方运算,例如

```
x=[1 2 3];y=[4 5 6];
z=x.^y
z=
     1    32   729
```

指数可以是标量。例如

```
x=[1  2  3];
z=x.^2
z=
     1  4  9
```

底也可以是标量。例如

```
x=[1  2  3];y=[4  5  6];
z=2.^[x y]
z=
     2     4     8    16    32    64
```

点运算在 MATLAB 中起着很重要的作用,也是许多初学者容易弄混的一个问题。下面再举一个例子进行说明。

当 $x=0.1,0.4,0.7,1$ 时,分别求 $y=\sin(x)\cos(x)$的值。命令应当写成

```
x=0.1:0.3:1;
y=sin(x).*cos(x);
```

其中求 y 的表达式中必须是点乘运算。运算过程是这样的:x 是含有 4 个元素的向量,sin(x)也

是含有 4 个元素的向量,各元素即对应 x 点的正弦函数值,同理 cos(x) 也是含有 4 个元素的向量,各元素即对应 x 点的余弦函数值,需要求各 x 点对应的 $y = \sin(x)\cos(x)$,所以必须是 sin(x) 与 cos(x)对应元素做乘法运算,即点乘运算。当 x 只是一个点时,sin(x)和 cos(x)均是标量,故直接用乘法运算就可以了。

3. MATLAB 常用数学函数

MATLAB 提供了许多数学函数,函数的自变量规定为矩阵变量,运算法则是将函数逐项作用于矩阵的元素上,因而运算的结果是一个与自变量同维数的矩阵。表 2.3 列出了一些常用数学函数。

表 2.3　常用数学函数

函数名	含　义	函数名	含　义
sin	正弦函数	exp	自然指数函数
cos	余弦函数	pow2	2 的幂
tan	正切函数	abs	绝对值函数
asin	反正弦函数	angle	复数的幅角
acos	反余弦函数	real	复数的实部
atan	反正切函数	imag	复数的虚部
sinh	双曲正弦函数	conj	复数共轭运算
cosh	双曲余弦函数	rem	求余数或模运算
tanh	双曲正切函数	mod	模除求余
asinh	反双曲正弦函数	fix	向零方向取整
acosh	反双曲余弦函数	floor	不大于自变量的最大整数
atanh	反双曲正切函数	ceil	不小于自变量的最小整数
sqrt	平方根函数	round	四舍五入到最邻近的整数
log	自然对数函数	sign	符号函数
log10	常用对数函数	gcd	最大公因子
log2	以 2 为底的对数函数	lcm	最小公倍数

函数使用说明:

(1) abs 函数可以求实数的绝对值、复数的模、字符串的 ASCII 码值。

(2) 用于取整的函数有 fix、floor、ceil、round,要注意它们的区别。round 函数作四舍五入用。设 a 为正整数,则其余 3 个函数的区别可表示为图 2.3。

设 $x = 2.45$,则 fix(x)、floor(x)、ceil(x)、round(x)的结果分别是 2,2,3,2。又设 $x = -2.65$,则 fix(x)、floor(x)、ceil(x)、round(x)的结果分别是 -2, -3, -2, -3。

(3) 函数 rem 与 mod 的区别。rem(x,y)和 mod(x,y)都要求 x,y 必须为相同大小的实矩阵或为标量。当 $y \neq 0$ 时,rem(x,y) = x - y. * fix(x./y),而 mod(x,y) = x - y. * floor(x./y),当 $y = 0$

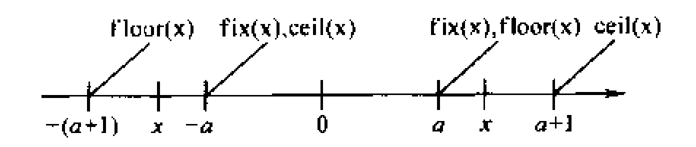


图 2.3 取整函数的区别

时，rem(x,0) = NaN，而 mod(x,0) = x。显然，当 x,y 同号时，rem(x,y)与 mod(x,y)相等。rem(x,y)的符号与 x 相同，而 mod(x,y)的符号与 y 相同。设 $x=5,y=3$，则 rem(x,y)和 mod(x,y)的结果都是 2。又设 $x=-5,y=3$，则 rem(x,y)和 mod(x,y)的结果分别是 −2 和 1。

2.4.2 关系运算

MATLAB 提供了 6 种关系运算符：<（小于）、<=（小于或等于）、>（大于）、>=（大于或等于）、==（等于）、~=（不等于）。它们的含义不难理解，但要注意其书写方法与数学中的不等式符号不尽相同。

关系运算符的运算法则为：

(1) 当两个比较量是标量时，直接比较两数的大小。若关系成立，关系表达式结果为 1，否则为 0。

(2) 当参与比较的量是两个维数相同的矩阵时，比较是对两矩阵相同位置的元素按标量关系运算规则逐个进行，并给出元素比较结果。最终的关系运算结果是一个维数与原矩阵相同的矩阵，它的元素由 0 或 1 组成。

(3) 当参与比较的一个是标量，而另一个是矩阵时，则把标量与矩阵的每一个元素按标量关系运算规则逐个比较，并给出元素比较结果。最终的运算结果是一个维数与矩阵相同的矩阵，它的元素由 0 或 1 组成。

例 2.4 产生5阶随机方阵 ***A***，其元素为[10,90]区间的随机整数，然后判断 ***A*** 的元素是否能被 3 整除。

(1) 生成 5 阶随机方阵 ***A***，命令和运行结果如下：

```
A = fix((90 - 10 + 1) * rand(5) + 10)
A =
      24     35     13     22     63
      23     39     47     80     80
      90     41     80     29     10
      45     57     85     62     21
      37     19     31     88     76
```

(2) 判断 ***A*** 的元素是否可以被 3 整除，命令和运行结果如下：

```
P = rem(A,3) == 0
P =
      1     0     0     0     1
      0     1     0     0     0
      1     0     0     0     0
```

```
     1     1     0     0     1
     0     0     0     0     0
```

其中,rem(A,3)是矩阵 **A** 的每个元素除以 3 的余数矩阵。此时,0 被扩展为与 **A** 同维数的零矩阵,**P** 是进行 == 比较的结果矩阵。

2.4.3 逻辑运算

MATLAB 提供了 3 种逻辑运算符:&(与)、|(或)和 ~(非)。

运算法则为:

(1) 在逻辑运算中,确认非零元素为真,用 1 表示,零元素为假,用 0 表示。

(2) 设参与逻辑运算的是两个标量 a 和 b,那么

a&b　a,b 全为非零时,运算结果为 1,否则为 0。

a|b　a,b 中只要有一个非零,运算结果为 1。

~a　当 a 是零时,运算结果为 1;当 a 非零时,运算结果为 0。

(3) 若参与逻辑运算的是两个同维矩阵,那么运算将对矩阵相同位置上的元素按标量规则逐个进行。最终运算结果是一个与原矩阵同维的矩阵,其元素由 1 或 0 组成。

(4) 若参与逻辑运算的一个是标量,一个是矩阵,那么运算将在标量与矩阵中的每个元素之间按标量规则逐个进行。最终运算结果是一个与矩阵同维的矩阵,其元素由 1 或 0 组成。

(5) 逻辑非是单目运算符,也服从矩阵运算规则。

(6) 在算术、关系、逻辑运算中,算术运算优先级最高,逻辑运算优先级最低。

例 2.5　在$[0,3\pi]$区间,求 $y=\sin(x)$的值。要求:

(1) 消去负半波,即$(\pi,2\pi)$区间内的函数值置 0。

(2) $\left(\frac{\pi}{3},\frac{2\pi}{3}\right)$和$\left(\frac{7\pi}{3},\frac{8\pi}{3}\right)$区间内取值均为 $\sin\frac{\pi}{3}$。

先根据自变量向量 x 产生函数值向量 y,然后按要求对 y 进行处理。处理的思路有两个:一是从自变量着手进行处理,二是从函数值着手进行处理。

方法 1:

```
x=0:pi/100:3*pi;y=sin(x);
y1=(x<pi|x>2*pi).*y;              %消去负半波
q=(x>pi/3&x<2*pi/3)|(x>7*pi/3&x<8*pi/3);
qn=~q;
y2=q*sin(pi/3)+qn.*y1;            %按要求处理第(2)步
```

方法 2:

```
x=0:pi/100:3*pi;y=sin(x);
y1=(y>=0).*y;                     %消去负半波
p=sin(pi/3);
y2=(y>=p)*p+(y<p).*y1;            %按要求处理第(2)步
```

此例说明,由于 MATLAB 以 0 或 1 表示关系运算和逻辑运算的结果,所以巧妙利用关系运算和逻辑运算能对函数值进行分段处理,即不需条件判断就能求分段函数的值。以本题的处理结果作为绘图数据,可以得到削顶的正弦半波曲线。

此外,MATLAB 还提供了一些关系与逻辑运算函数,表 2.4 列出常用的几个。

表 2.4 关系与逻辑运算函数

函数名	含 义
all	若向量的所有元素非零,则结果为 1
any	向量中任何一个元素非零,都给出结果 1
exist	检查变量在工作空间是否存在,若存在,则结果为 1,否则为 0
find	找出向量或矩阵中非零元素的位置
isempty	若被查变量是空阵,则结果为 1
isglobal	若被查变量是全局变量,则结果为 1
isinf	若元素是 ± inf,则结果矩阵相应位置元素取 1,否则取 0
isnan	若元素是 nan,则结果矩阵相应位置元素取 1,否则取 0
isfinite	若元素值大小有限,则结果矩阵相应位置元素取 1,否则取 0
issparse	若变量是稀疏矩阵,则结果矩阵相应位置元素取 1,否则取 0
isstr	若变量是字符串,则结果矩阵相应位置元素取 1,否则取 0
xor	若两矩阵对应元素同为 0 或非 0,则结果矩阵相应位置元素取 0,否则取 1

2.5 字 符 串

在 MATLAB 语言中,字符串是用单撇号括起来的字符序列。例如

```
xm = ' Central South University '
```

输出结果是:

```
xm =
     Central South University
```

MATLAB 将字符串当作一个行向量,每个元素对应一个字符,其标识方法和数值向量相同。也可以建立多行字符串矩阵。例如

```
ch = [' abcdef ';' 123456 '];
```

这里要求各行字符数要相等。为此,有时不得不用空格来调节各行的长度,使它们彼此相等。

字符串是以 ASCII 码形式存储的。abs 和 double 函数都可以用来获取字符串矩阵所对应的 ASCII 码数值矩阵。相反,char 函数可以把 ASCII 码矩阵转换为字符串矩阵。

例 2.6 建立一个字符串向量,然后对该向量做如下处理:

(1) 取第 1～5 个字符组成的子字符串。

(2) 将字符串倒过来重新排列。

(3) 将字符串中的小写字母变成相应的大写字母,其余字符不变。

(4) 统计字符串中小写字母的个数。

命令及运行结果如下：

```
ch = 'ABc123d4e56Fg9';
subch = ch(1:5)             %取子字符串
subch =
      ABc12
revch = ch(end:-1:1)        %将字符串倒排
revch =
     9gF65e4d321cBA
k = find(ch >= 'a' &ch <= 'z');         %找小写字母的位置
ch(k) = ch(k) - ('a' - 'A');            %将小写字母变成相应的大写字母
char(ch)
ans =
        ABC123D4E56FG9
length(k)                               %统计小写字母的个数
ans =
       4
```

与字符串有关的另一个重要函数是 eval，其调用格式为：

```
eval(t)
```

其中 t 为字符串。它的作用是把字符串的内容作为对应的 MATLAB 语句来执行。例如

```
t = pi;
m = '[t,sin(t),cos(t)]';
y = eval(m)
y =
     3.1416    0.0000    -1.0000
```

MATLAB 还有许多与字符串处理有关的函数，表 2.5 列出几个常用的函数。

表 2.5　字符串处理函数

函数名	含　义	函数名	含　义
setstr	将 ASCII 码值转换成字符	str2num	将字符串转换成数值
mat2str	将矩阵转换成字符串	strcat	用于字符串的连接
num2str	将数值转换成字符串	strcmp	用于字符串的比较
int2str	将整数转换成字符串		

关于字符串的写法，还要注意两点：

(1) 若字符串中的字符含有单撇号，则该单撇号字符应用两个单撇号来表示。例如

```
disp('I''m a teacher.')
```

将输出：

```
    I'm a teacher.
```

(2) 对于较长的字符串可以用字符串向量表示，即用[ ]括起来。例如

```
f = 70;
```

```
c = (f - 32)/1.8;
disp([' Room temperature is ',num2str(c),' degrees C.'])
```

其中 disp 函数的自变量是一个长字符串。输出为:

```
Room temperature is 21.11 degrees C.
```

2.6 结构和单元

2.6.1 结构数据

MATLAB 通过使用结构(Structure)数据类型把一组不同类型的数据同时又是在逻辑上相关的数据组成一个有机的整体,以便于管理和引用。比如要存储学生基本情况数据就可采用结构数据。

1. 结构矩阵的建立与引用

结构矩阵的元素可以是不同的数据类型,它能将一组具有不同属性的数据纳入到一个统一的变量名下进行管理。建立一个结构矩阵可采用给结构成员赋值的办法。具体格式为:

结构矩阵名.成员名 = 表达式

其中表达式应理解为矩阵表达式。例如,建立含有 3 个元素的结构矩阵 a

```
a(1).x1 = 10; a(1).x2 = ' liu '; a(1).x3 = [11,21;34,78];
a(2).x1 = 12; a(2).x2 = ' wang '; a(2).x3 = [34,191;27,578];
a(3).x1 = 14; a(3).x2 = ' cai '; a(3).x3 = [13,890;67,231];
```

结构矩阵元素的成员也可以是结构数据。例如

```
a(2).x1.x11 = 90; a(2).x1.x12 = 12; a(2).x1.x13 = 30;
```

以上建立的结构矩阵 *a* 含有 3 个元素,每个元素又含有 3 个成员,成员 a(2).x1 又是含有 3 个成员的结构数据。对结构数据的引用,可以引用其成员,也可以引用结构矩阵的元素或结构变量。例如

```
a(2).x3          %引用矩阵元素 a(2)的成员 x3
ans =
    34    191
    27    578
a(2)             %引用矩阵元素 a(2)
ans =
    x1:[1x1 struct]
    x2:' wang '
    x3:[2x2 double]
a                %引用结构矩阵 a
a =
1x3 struct array with fields:
    x1
    x2
```

```
x3
```

引用结构矩阵元素的成员时，显示其值。引用结构矩阵元素时，显示成员名和它的值，但成员是矩阵时，不显示其具体内容，只显示成员矩阵大小参数。引用结构矩阵时，只显示结构矩阵的大小参数和成员名。

2. 结构成员的修改

可以根据需要增加或删除结构的成员。例如要给结构矩阵 *a* 增加一个成员 *x*4，可给 a 中任意一个元素增加成员 *x*4

```
a(1).x4 = '410075';
```

但其他成员均为空矩阵，可以使用赋值语句给它赋确定的值。

要删除结构的成员，则可以使用 rmfield 函数来完成。例如，删除成员 *x*4

```
a = rmfield(a,'x4');
```

3. 关于结构的函数

关于结构的函数见表 2.6。

表 2.6　关于结构的函数

函数名	含　义	函数名	含　义
struct	建立或转换为结构矩阵	fieldnames	获取结构成员名
getfield	获取结构成员的内容	setfield	设定结构成员的内容
rmfield	删除结构成员	isfield	成员在结构中时，值为真
isstruct	是结构时，值为真		

2.6.2　单元数据

单元(Cell)数据的概念与结构有些类似，也是把不同属性的数据放在一个变量中。所不同的是，结构变量的各个元素下有成员，每个成员都有自己的名字，对成员的引用是：结构变量名.成员名。而单元矩阵的各个元素就是不同类型的数据，用带有大括号下标的形式引用单元矩阵元素。

1. 单元矩阵的建立与引用

建立单元矩阵和一般矩阵相似，只是矩阵元素用大括号括起来。如建立单元矩阵 b

```
b = {10,'liu',[11,21;34,78];12,'wang',[34,191;27,578];14,'cai',[13,890;67,231]}
b =
    [10]    'liu'     [2x2 double]
    [12]    'wang'    [2x2 double]
    [14]    'cai'     [2x2 double]
```

可以用带有大括号下标的形式引用单元矩阵元素。例如 b{3,3}。单元矩阵的元素可以是结构或单元数据。例如，先建立结构变量 *y*，给上面建立的单元矩阵 ***b*** 的元素 b{3,4}赋值：

```
y.x1 = 34; y.x2 = 56;
b{3,4} = y;
```

可以使用 celldisp 函数来显示整个单元矩阵。如 celldisp(b)。还可以删除单元矩阵中的某个

元素。如删除 *b* 的第 3 个元素

```
b(3) = []
b =
  Columns 1 through 7
    [10]    [12]    'liu'    'wang'    'cai'    [2x2 double]    [2x2 double]
  Columns 8 through 11
    [2x2 double]    []    []    [1x1 struct]
```

单元矩阵 *b* 的第 3 个元素被删除后，*b* 变成行向量。注意，这里是 b(3)，而不是 b{3}。b{3} = []是将 *b* 的第 3 个元素置为空矩阵，而不是删除它。

2. 关于单元的函数

关于单元的函数见表 2.7。

表 2.7 关于单元的函数

函数名	作　用	函数名	作　用
celldisp	显示单元矩阵内容	cellplot	显示单元矩阵的图形描述
num2cell	把数字矩阵转换为单元矩阵	deal	把输入分配给输出
cell2struct	把单元矩阵转换为结构矩阵	struct2cell	把结构矩阵转换为单元矩阵
iscell	是单元矩阵时，值为真		

习 题 二

1. 如何理解"矩阵是 MATLAB 最基本的数据对象"？

2. 设 A 和 B 是两个同维同大小的矩阵，问：

(1) A * B 和 A . * B 的值是否相等？

(2) A./B 和 B. \ A 的值是否相等？

(3) A/B 和 B \ A 的值是否相等？

(4) A/B 和 B \ A 所代表的数学含义是什么？

3. 写出完成下列操作的命令。

(1) 建立 3 阶单位矩阵 *A*。

(2) 建立 5×6 随机矩阵 *A*，其元素为[100,200]范围内的随机整数。

(3) 将矩阵 *A* 第 2～5 行中第 1,3,5 列元素赋给矩阵 *B*。

(4) 删除矩阵 *A* 的第 7 号元素。

(5) 将矩阵 *A* 的每个元素值加 30。

(6) 求矩阵 *A* 的大小和维数。

(7) 将向量 *t* 的 0 元素用机器零来代替。

(8) 将含有 12 个元素的向量 *x* 转换成 3×4 矩阵。

(9) 求一个字符串的 ASCII 码。

(10) 求一个 ASCII 码所对应的字符。

4. 当 x = [-5,7,15,3], y = [3,-4,5,0]时，试分析 rem(x,y)和 mod(x,y)的执行结果。

5. 下列命令执行后，L1,L2,L3,L4 的值分别是多少？

A = 1:9; B = 10 - A;

L1 = A == B;

L2 = A <= 5;

L3 = A > 3&A < 7;

L4 = find(A > 3&A < 7);

6. 当 A = [34, NaN, Inf, -Inf, -pi, eps, 0]时，分析下列函数的执行结果：all(A), any(A), isnan(A), isinf(A), isfinite(A)。

7. 用结构矩阵来存储 5 名学生的基本情况数据，每名学生的数据包括学号、姓名、专业和 6 门课程的成绩。

8. 建立单元矩阵 ***B*** 并回答有关问题。

B{1,1} = 1;

B{1,2} = ' Brenden ';

B{2,1} = reshape(1:9,3,3);

B{2,2} = {12,34,2;54,21,3;4,23,67};

(1) size(B)和 ndims(B)的值分别是多少？

(2) B(2)和 B(4)的值分别是多少？

(3) B(3) = []和 B{3} = []执行后，B 的值分别是多少？

第 3 章 MATLAB 程序设计

MATLAB 命令有两种执行方式:一种是交互式的命令执行方式,另一种是 M 文件的程序执行方式。在命令执行方式下,用户在命令窗口逐条输入命令,MATLAB 逐条解释执行。这种方式虽然操作简单、直观,但速度慢,执行过程不能保留。当某些操作需反复进行时,更使人感到单调厌烦。在程序执行方式下,用户将有关命令编成程序存储在一个文件中(称为 M 文件),当运行该程序后,MATLAB 就会自动依次执行该文件中的命令,直至全部命令执行完毕。以后需要这些命令时,只需再次运行该程序。可见,程序执行方式大大减少了用户的重复劳动,成为实际应用中的重要执行方式。

本章介绍 M 文件、数据的输入输出和有关控制语句,在此基础上,再介绍 MATLAB 程序设计的基本方法。

3.1 M 文 件

3.1.1 M 文件的建立与编辑

M 文件是一个文本文件,它可以用任何编辑程序来建立和编辑,而一般常用且最为方便的是使用 MATLAB 提供的文本编辑器。

1. 建立新的 M 文件

为建立新的 M 文件,启动 MATLAB 文本编辑器有 3 种方法:

(1) 菜单操作。从 MATLAB 命令窗口的 File 菜单中选择 New 菜单项,再选择 M - file 命令,屏幕将出现 MATLAB Editor/Debugger 窗口(如图 3.1 所示)。MATLAB Editor/Debugger 是一个集编辑与调试两种功能于一体的工具环境。利用它不仅可以完成基本的文本编辑操作,还可以对 M 文件进行调试。MATLAB 文本编辑器的操作界面与使用方法和其他 Windows 编辑器相似,这里不作详细介绍。

启动 MATLAB 文本编辑器后,在文档窗口中输入 M 文件的内容,输入完毕后,选择文本编辑器窗口 File 菜单的 Save 或 Save As 命令存盘。注意,M 文件存放的位置一般是 MATLAB 缺省的用户工作目录 c:\matlabr11\work,当然也可以是别的目录。如果是别的目录,则应该将该目录设定为当前目录或将其加到搜索路径中。

(2) 命令操作。在 MATLAB 命令窗口输入命令 edit,启动 MATLAB 文本编辑器后,输入 M 文件的内容并存盘。

(3) 命令按钮操作。单击 MATLAB 命令窗口工具栏上的新建命令按钮,启动 MATLAB 文本编辑器后,输入 M 文件的内容并存盘。



图3.1 MATLAB Editor/Debugger 窗口

2. 编辑已有的M文件

编辑已有的M文件，也有3种方法：

(1) 菜单操作。从MATLAB命令窗口的File菜单中选择Open M－file命令，则屏幕出现Open对话框，在Open对话框中选中所需打开的M文件。在文档窗口可以对打开的M文件进行编辑修改。编辑完成后，将M文件存盘。

(2) 命令操作。在MATLAB命令窗口输入命令：edit 文件名，则打开指定的M文件。

(3) 命令按钮操作。单击MATLAB命令窗口工具栏上的打开命令按钮，再从弹出的对话框中选择所需打开的M文件。

3.1.2 M文件的分类

M文件有两类：命令文件(Script File)和函数文件(Function File)。它们的扩展名均为.m，主要区别在于：

(1) 命令文件没有输入参数，也不返回输出参数，而函数文件可以带输入参数，也可返回输出参数。

(2) 命令文件对MATLAB工作空间中的变量进行操作，文件中所有命令的执行结果也完全返回到工作空间中，而函数文件中定义的变量为局部变量，当函数文件执行完毕时，这些变量被清除。

(3) 命令文件可以直接运行，在MATLAB命令窗口输入命令文件的名字，就会顺序执行命令文件中的命令，而函数文件不能直接运行，而要以函数调用的方式来调用它。

例3.1 建立一个命令文件将变量 a，b 的值互换，然后运行该命令文件。

首先建立命令文件并以文件名exch.m存盘

```
clear;
```

```
a = 1:10; b = [11,12,13,14;15,16,17,18];
c = a;a = b;b = c;
a
b
```

然后在 MATLAB 的命令窗口中输入 exch,将会执行该命令文件,输出为:

```
a =
    11    12    13    14
    15    16    17    18
b =
    1    2    3    4    5    6    7    8    9    10
```

调用该命令文件时,不用输入参数,也没有输出参数,文件自身建立需要的变量。当文件执行完毕后,可以用命令 whos 查看工作空间中的变量。这时会发现 a,b,c 仍然保留在工作空间中。

例 3.2 建立一个函数文件将变量 a,b 的值互换,然后在命令窗口调用该函数文件。

首先建立函数文件 fexch.m

```
function [a,b] = exch(a,b)
c = a;a = b;b = c;
```

然后在 MATLAB 的命令窗口调用该函数文件

```
clear;
x = 1:10; y = [11,12,13,14;15,16,17,18];
[x,y] = fexch(x,y)
```

输出为:

```
x =
    11    12    13    14
    15    16    17    18
y =
    1    2    3    4    5    6    7    8    9    10
```

调用该函数文件时,既有输入参数,又有输出参数。当函数调用完毕后,可以用命令 whos 查看工作空间中的变量。这时会发现 a,b,c 未被保留在工作空间中。

3.2 数据的输入输出

MATLAB 的输入输出方式包括从命令窗口的输入输出以及通过文件操作的输入输出。此外,为了顺应现代软件开发技术的潮流,MATLAB 还提供图形界面的输入输出方式。本节先介绍命令窗口的输入输出,关于文件操作和图形界面设计分别在 3.7 节和第 7 章介绍 。

3.2.1 input 函数

MATLAB 提供了一些输入输出函数,允许用户和计算机之间进行数据交换。如果用户想从键盘输入数据,则可以使用 input 函数来进行,该函数的调用格式为:

```
A = input(提示信息,选项);
```

其中提示信息为一个字符串,用于提示用户输入什么样的数据。例如,从键盘输入 *A* 矩阵,可以采用下面的命令来完成:

```
A = input('输入 A 矩阵:');
```

执行该语句时,首先在屏幕上显示提示信息"输入 A 矩阵:",然后等待用户从键盘按 MATLAB 规定的格式输入 *A* 矩阵的值。

如果在 input 函数调用时采用' s '选项,则允许用户输入一个字符串。例如,想输入一个人的姓名,可采用命令:

```
xm = input(' What" s your name? ',' s ')
```

3.2.2 disp 函数

MATLAB 提供的命令窗口输出函数主要有 disp 函数,其调用格式为:

```
disp(输出项)
```

其中输出项既可以为字符串,也可以为矩阵。例如

```
A = ' Hello, Brenden ';
disp(A)
```

输出为:

```
Hello, Brenden
```

又如

```
A = [1, 2, 3; 4, 5, 6; 7, 8,9];
disp(A)
```

输出为:

```
1   2   3
4   5   6
7   8   9
```

注意,和前面介绍的矩阵显示方式不同,用 disp 函数显示矩阵时将不显示矩阵的名字,而且其格式更紧密,且不留任何没有意义的空行。

例 3.3 求一元二次方程 $ax^2 + bx + c = 0$ 的根。

由于 MATLAB 能进行复数运算,所以不需要进行判别式的判断。程序如下:

```
a = input(' a = ? ');
b = input(' b = ? ');
c = input(' c = ? ');
d = b * b - 4 * a * c;
x = [( - b + sqrt(d))/(2 * a),( - b - sqrt(d))/(2 * a)];
disp([' x1 = ',num2str(x(1)),',x2 = ',num2str(x(2))]);
```

第一次运行:

```
a = ? 5
b = ? 67
c = ? 34
x1 = -0.52829,x2 = -12.8717
```

第二次运行：

```
a = ? 34
b = ? 23
c = ? 45
x1 = -0.33824 + 1.0996i,x2 = -0.33824 - 1.0996i
```

3.2.3 pause 函数

当程序运行时，为了查看程序的中间结果或者观看输出的图形，有时需要暂停程序的执行。这时可以使用 pause 函数，其调用格式为：

```
pause(延迟秒数)
```

如果省略延迟时间，直接使用 pause，则将暂停程序，直到用户按任一键后程序继续执行。

若要强行中止程序的运行可使用 Ctrl + C 命令。

3.3 选择结构

选择结构是根据给定的条件成立或不成立，分别执行不同的语句。MATLAB 用于实现选择结构的语句有 if 语句、switch 语句和 try 语句。

3.3.1 if 语句

在 MATLAB 中，if 语句有 3 种格式。

1. 单分支 if 语句

语句格式为：

```
if 条件
   语句组
end
```

当条件成立时，则执行语句组，执行完之后继续执行 if 语句的后继语句，若条件不成立，则直接执行 if 语句的后继语句。例如，当 x 是整数矩阵时，输出 x 的值。语句如下：

```
if fix(x) == x
   disp(x);
end
```

2. 双分支 if 语句

语句格式为：

```
if 条件
   语句组 1
else
   语句组 2
end
```

当条件成立时，执行语句组 1，否则执行语句组 2，语句组 1 或语句组 2 执行后，再执行 if 语句的后继语句。

例 3.4　计算分段函数

$$\begin{cases} \cos(x+1)+\sqrt{x2+1}, & x=10 \\ x\sqrt{x+\sqrt{x}}, & x\neq 10 \end{cases}$$

程序如下：

```
x = input('请输入 x 的值:');
if x == 10
    y = cos(x + 1) + sqrt(x * x + 1);
else
    y = x * sqrt(x + sqrt(x));
end
y
```

也可以用单分支 if 语句来实现：

```
x = input('请输入 x 的值:');
y = cos(x + 1) + sqrt(x * x + 1);
if  x~ = 10
    y = x * sqrt(x + sqrt(x));
end
y
```

或用以下程序：

```
x = input('请输入 x 的值:');
if x == 10
    y = cos(x + 1) + sqrt(x * x + 1);
end
if x ~ = 10
    y = x * sqrt(x + sqrt(x));
end
y
```

3. 多分支 if 语句

语句格式为：

```
if  条件 1
    语句组 1
elseif  条件 2
    语句组 2
    …
elseif  条件 m
        语句组 m
else
    语句组 m + 1
```

```
end
```

语句执行过程如图 3.2 所示，可用于实现多分支选择结构。

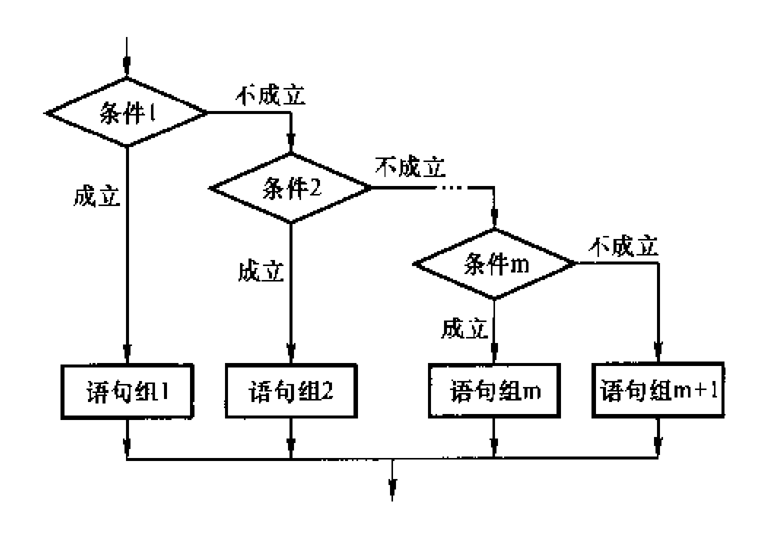


图 3.2　多分支 if 语句的执行过程

例 3.5　输入一个字符，若为大写字母，则输出其后继字符，若为小写字母，则输出其前导字符，若为数字字符则输出其对应的数值，若为其他字符则原样输出。

关于字符的处理，用 abs 或 double 函数可得到一个字符的 ASCII 码，用 char 和 setstr 函数可得到 ASCII 码对应的字符。程序如下：

```
c = input('请输入一个字符','s');
if c> ='A' & c< ='Z'
    disp(setstr(abs(c) + 1));
elseif c> ='a' & c< ='z'
    disp(setstr(abs(c) - 1));
elseif c> ='0' & c< ='9'
    disp(abs(c) - abs('0'));
else
    disp(c);
end
```

3.3.2　switch 语句

switch 语句根据表达式的取值不同，分别执行不同的语句。其语句格式为：

```
switch 表达式
   case 表达式 1
     语句组 1
   case 表达式 2
     语句组 2
       …
   case 表达式 m
```

```
        语句组 m
    otherwise
        语句组 m+1
end
```

switch 语句的执行过程如图 3.3 所示。当表达式的值等于表达式 1 的值时，执行语句组 1，当表达式的值等于表达式 2 的值时，执行语句组 2、…，当表达式的值等于表达式 m 的值时，执行语句组 m，当表达式的值不等于 case 所列的表达式的值时，执行语句组 m+1。当任一分支的语句执行完后，直接执行 switch 语句的下一句。

switch 子句后面的表达式应为一个标量或一个字符串，case 子句后面的表达式不仅可以为一个标量或一个字符串，而且还可以为一个单元矩阵。如果 case 子句后面的表达式为一个单元矩阵，则表达式的值等于该单元矩阵中的某个元素时，执行相应的语句组。

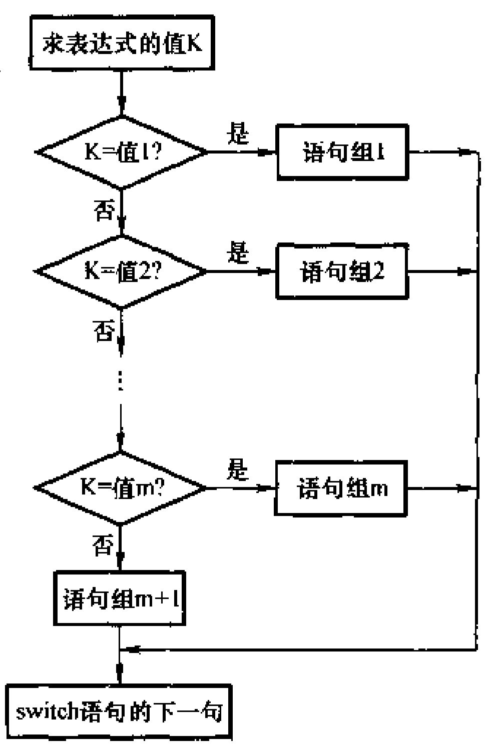


图 3.3　switch 语句的执行过程

例 3.6　某商场对顾客所购买的商品实行打折销售，标准如下（商品价格用 price 来表示）：

price < 200	没有折扣
200 ≤ price < 500	3% 折扣
500 ≤ price < 1000	5% 折扣
1000 ≤ price < 2500	8% 折扣
2500 ≤ price < 5000	10% 折扣
5000 ≤ price	14% 折扣

求所售商品的实际销售价格。

程序如下：

```
price = input('请输入商品价格');
switch fix(price/100)
  case {0,1}
    rate = 0;
  case {2,3,4}
    rate = 3/100;
  case num2cell(5:9)
    rate = 5/100;
  case num2cell(10:24)
    rate = 8/100;
  case num2cell(25:49)
    rate = 10/100;
  otherwise
    rate = 14/100;
  end
  price = price * (1 - rate)
```

3.3.3 try 语句

MATLAB 从 5.2 版开始提供了 try 语句,这是一种试探性执行语句。语句格式为:

```
try
  语句组 1
catch
  语句组 2
end
```

try 语句先试探性执行语句组 1,如果语句组 1 在执行过程中出现错误,则将错误信息赋给保留的 lasterr 变量,并转去执行语句组 2。这种试探性执行语句是其他高级语言所没有的。

例 3.7 矩阵乘法运算要求两矩阵的维数相容,否则会出错。先求两矩阵的乘积,若出错,则自动转去求两矩阵的点乘。

程序如下:

```
A = [1,2,3;4,5,6]; B = [7,8,9;10,11,12];
try
  C = A * B;
catch
  C = A. * B;
end
C
lasterr                          %显示出错原因
```

输出结果为:

```
C =
    7     16     27
    40     55     72
ans =
Error using  = = >  *
Inner matrix dimensions must agree.
```

3.4 循环结构

循环是指按照给定的条件,重复执行指定的语句,这是十分重要的一种程序结构。MATLAB 提供了两种实现循环结构的语句:for 语句和 while 语句。

3.4.1 for 语句

语句格式为:

```
for 循环变量 = 表达式 1:表达式 2:表达式 3
  循环体语句
```

```
end
```

其中表达式 1 的值为循环变量的初值，表达式 2 的值为步长，表达式 3 的值为循环变量的终值。步长为 1 时，表达式 2 可以省略。

for 语句的执行过程如图 3.4 所示。首先计算 3 个表达式的值，再将表达式 1 的值赋给循环变量，如果此时循环变量的值介于表达式 1 和表达式 3 的值之间，则执行循环体语句，否则结束循环的执行。执行完一次循环之后，循环变量自增一个表达式 2 的值，然后再判断循环变量的值是否介于表达式 1 和表达式 3 之间，如果满足，仍然执行循环体，直至不满足为止。这时将结束 for 语句的执行，而继续执行 for 语句后面的语句。

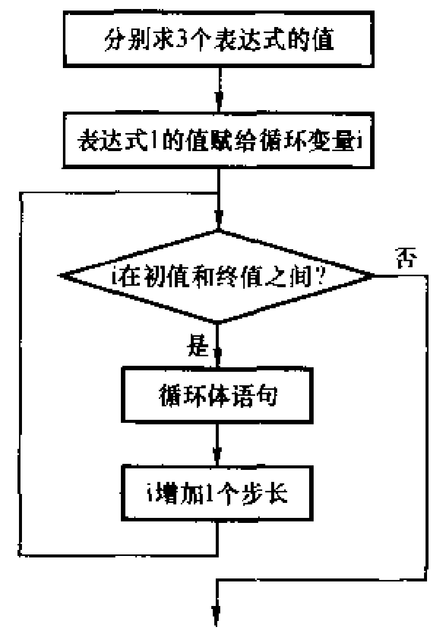


图 3.4　for 语句执行过程

例 3.8　已知，$y=\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}$，当 $n=100$ 时，求 y 的值。

程序如下：

```
y = 0;n = 100;
for i = 1:n
  y = y + 1/i/i;
end
y
```

在实际 MATLAB 编程中，为提高程序的执行速度，常用向量运算来代替循环操作，所以上述程序通常由下面的程序来代替：

```
n = 100; i = 1:n;
f = 1./i.^2;
y = sum(f)
```

在这一程序中，首先生成一个向量 i，然后用 i 生成向量 f，f 各元素值即对应于 y 的各累加项，再用 MATLAB 提供的 sum 函数求 f 各个元素之和。如果程序中的 n 值由 100 改成 10000，再分别运行这两个程序，则可以明显地看出，后一种方法编写的程序比前一种方法快得多。

例 3.9　设，$f(x)=e^{-0.5x}\sin\left(x+\frac{\pi}{6}\right)$，求 $s=\int_0^{3\pi}f(x)dx$。

求函数 $f(x)$ 在 $[a,b]$ 上的定积分，其几何意义就是求曲线 $y=f(x)$ 与直线 $x=a,x=b,y=0$ 所围成的曲边梯形的面积。为了求得曲边梯形面积，先将积分区间 $[a,b]$ 分成 n 等分，每个区间的宽度为 $h=(b-a)/n$，对应地将曲边梯形分成 n 等分，每个小部分即是一个小曲边梯形。近似求出每个小曲边梯形面积，然后将 n 个小曲边梯形的面积加起来，就得到总面积，即定积分的近似值。近似地求每个小曲边梯形的面积，常用的方法有：矩形法、梯形法以及辛普生法等。以梯形法为例，程序如下：

```
a = 0;b = 3 * pi;n = 1000; h = (b - a)/n;
x = a; s = 0; f0 = exp( - 0.5 * x) * sin(x + pi/6);
for i = 1:n
      x = x + h;
      f1 = exp( - 0.5 * x) * sin(x + pi/6);
```

```
        s = s + (f0 + f1) * h/2;
        f0 = f1;
    end
    s
```

上述程序来源于传统的编程思想。也可以利用向量运算,从而使得程序更加简洁,更赋有MATLAB的特点。程序如下:

```
a = 0;b = 3 * pi;n = 1000; h = (b - a)/n;
x = a:h:b; f = exp( - 0.5 * x) . * sin(x + pi/6);
for i = 1:n
    s(i) = (f(i) + f(i + 1)) * h/2;
end
s = sum(s)
```

程序中 x,f,s 均为向量,f 的元素为各个 x 点的函数值,s 的元素分别为 n 个梯形的面积,s 各元素之和即定积分近似值($s = 0.9008$)。

事实上,MATLAB 提供了有关数值积分的标准函数,实际应用中可直接调用这些函数求数值积分,这些内容将在第 5 章介绍。

在上述例子中,for 语句的循环变量都是标量,这与其他高级语言的相关循环语句(如 FORTRAN 语言中的 DO 语句,C 语言中的 for 语句等)等价。按照 MATLAB 的定义,for 语句的循环变量可以是一个列向量。for 语句更一般的格式为:

```
for 循环变量 = 矩阵表达式
   循环体语句
end
```

执行过程是依次将矩阵的各列元素赋给循环变量,然后执行循环体语句,直至各列元素处理完毕。实际上,"表达式 1:表达式 2:表达式 3"是一个仅为一行的矩阵(行向量),因而列向量是单个数据。所以本节一开始给出的 for 语句格式是一种特例。

例 3.10 已知5个学生4门功课的成绩,求每名学生的总成绩。

程序如下:

```
s = 0;
a = [65,76,56,78;98,83,74,85;76,67,78,79;98,58,42,73;67,89,76,87];
for k = a
   s = s + k;
end
disp(s ');
```

程序的最后一行用到了矩阵转置运算符,这里是将列向量转置为行向量。

3.4.2 while 语句

while 语句的一般格式为:

```
while (条件)
     循环体语句
end
```

其执行过程为：若条件成立，则执行循环体语句，执行后再判断条件是否成立，如果不成立则跳出循环（如图 3.5 所示）。

例 3.11　根据矩阵指数的幂级数展开式求矩阵指数函数值。

$$E^X = I + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \cdots + \frac{X^n}{n!} + \cdots$$

程序中，设 X 是给定的矩阵，E 是矩阵指数函数值，F 是展开式的项，n 是项数，循环一直进行到 F 很小，以至于 F 值加在 E 上对 E 的值影响不大时为止。为了判断 F 是否很小，可利用矩阵范数的概念。矩阵 A 的范数的一种定义是：$\max\limits_{1 \leqslant j \leqslant n} \sum\limits_{i=1}^{n} |a_{ij}|$，在 MATLAB 中用 norm(A,1)函数来计算。所以当 norm(F,1) = 0 时，认为 F 很小，应退出循环的执行。程序如下：

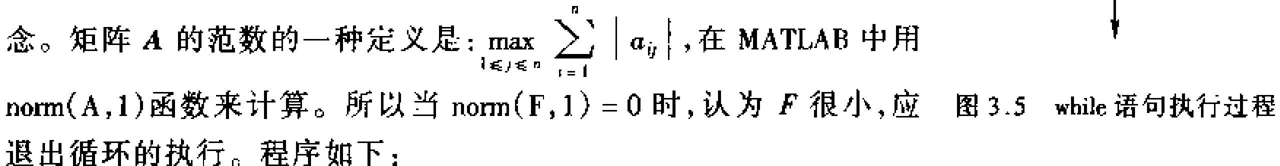


图 3.5　while 语句执行过程

```
X = input('X = ');
E = zeros(size(X));
F = eye(size(X));
n = 1;
while norm(F,1) > 0
   E = E + F;
   F = F * X/n;
   n = n + 1;
end
E
expm(X)                    %调用 MATLAB 矩阵指数函数求矩阵指数函数值
```

程序运行时，若从键盘输入 X 的值：[0.5,2,0;1,-1,-0.5;0.9,1,0.75]，则程序输出结果为：

```
E =
    2.6126    2.0579   -0.6376
    0.7420    0.7504   -0.5942
    2.5678    2.3359    1.5549
ans =
    2.6126    2.0579   -0.6376
    0.7420    0.7504   -0.5942
    2.5678    2.3359    1.5549
```

运行结果表明，程序运行结果与 MATLAB 矩阵指数函数 expm(X)的结果一致。本程序涉及到矩阵运算，初学者可能不太习惯。如能分析一下程序的执行过程，对领会编程思想是有益的。另外，我们知道矩阵乘法的交换律不成立，但这里要请读者分析一下程序中的语句 F = F * X/n 可否写成 F = X * F/n，为什么？

与循环结构相关的还有一个 break 语句，当在循环体内执行到该语句时，程序将跳出循环。该语句一般与 if 语句配合使用。

例 3.12　用 while 语句实现例 3.8。

程序如下：

```
y=0; i=1;
while 1
   f=1/i/i;
   y=y+f;
   if i==100
       break;
   end
   i=i+1;
end
y
```

在程序中，循环的条件为1，即循环条件总是满足的，这是一个永真循环。为了使循环能正常结束，在循环体中加了一个if语句，当 i 等于100时，执行break命令，从而跳出for循环。

3.4.3 循环的嵌套

如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为多重循环结构。实现多重循环结构仍用前面介绍的3种循环语句。因为任一循环语句的循环体部分都可以包含另一个循环语句，这种循环语句的嵌套为实现多重循环提供了方便。

多重循环的嵌套层数可以是任意的。可以按照嵌套层数，分别叫做二重循环、三重循环等。处于内部的循环叫作内循环，处于外部的循环叫作外循环。

在设计多重循环时，要特别注意内、外循环之间的关系，以及各语句放置的位置，不要搞错。

例3.13 用筛选法求某正整数范围内的全部素数。

素数是大于1且除了1和它本身以外不能被其他任何整数所整除的整数。用筛选法求素数的基本思想是：要找出2～m之间的全部素数，首先在2～m中划去2的倍数(不包括2)，然后划去3的倍数(不包括3)，由于4已被划去，再找5的倍数（不包括5)，…，直到再划去不超过$\sqrt{m}$的数的倍数，剩下的数都是素数。程序如下：

```
m=input('m=');
p=1:m; p(1)=0;
for i=2:sqrt(m)
    for j=2*i:i:m
       p(j)=0;
    end
end
n=find(p~=0);
p(n)
```

外循环控制 i 从2到$\sqrt{m}$变化，内循环在 p 中划去 i 的倍数(不包括 i)，p 中剩下的数都是素数。find函数找出 p 中非0元素的下标并赋予变量 n(注意，n 为向量)。

关于在 p 中划去 i 的倍数(不包括 i)，可利用矩阵运算一步完成，从而得到更为简洁的程序：

```
m=input('m=');
p=2:m;
```

```
for i = 2:sqrt(m)
    n = find(rem(p,i) == 0&p ~ = i);
    p(n) = [];
end
p
```

变量 n 为 p 中能被 i 整除而 i 不等于 p 的元素的下标,然后将 p 中该位置上的元素剔除,剩下的全为素数。

3.5 函数文件

函数文件是另一种形式的M文件,每一个函数文件都定义一个函数。事实上,MATLAB提供的标准函数大部分都是由函数文件定义的。

3.5.1 函数文件的基本结构

函数文件由function语句引导,其基本结构为:

```
function 输出形参表 = 函数名(输入形参表)
  注释说明部分
  函数体语句
```

其中以function开头的一行为引导行,表示该M文件是一个函数文件。函数名的命名规则与变量名相同。输入形参为函数的输入参数,输出形参为函数的输出参数。当输出形参多于一个时,则应该用方括号括起来。

说明:

(1) 关于函数文件名。函数文件名通常由函数名再加上扩展名.m组成,不过函数文件名与函数名也可以不相同。当两者不同时,MATLAB将忽略函数名而确认函数文件名,因此调用时使用函数文件名。不过最好把文件名和函数名统一,以免出错。

(2) 关于注释说明部分。注释说明包括三部分内容:

① 紧随函数文件引导行之后以%开头的第一注释行。这一行一般包括大写的函数文件名和函数功能简要描述,供lookfor关键词查询和help在线帮助用。

② 第一注释行及之后连续的注释行。通常包括函数输入输出参数的含义及调用格式说明等信息,构成全部在线帮助文本。

③ 与在线帮助文本相隔一空行的注释行。包括函数文件编写和修改的信息,如作者、修改日期、版本等内容,用于软件档案管理。

(3) 关于return语句。如果在函数文件中插入了return语句,则执行到该语句就结束函数的执行,程序流程转至调用该函数的位置。通常,在函数文件中也可不使用return语句,这时在被调函数执行完成后自动返回。

例3.14 编写函数文件求半径为 r 的圆的面积和周长。

函数文件如下:

```
function [s,p] = fcircle(r)
```

```
%CIRCLE   calculate the area and perimeter of a circle of radii r
%r             圆半径
%s             圆面积
%p             圆周长
%2001年7月30日编
s = pi * r * r;
p = 2 * pi * r;
```

将以上函数文件以文件名 fcircle.m 存入 c:\matlabr11\work 下,然后在 MATLAB 命令窗口调用该函数:

```
[s,p] = fcircle(10)
```

输出结果是:

```
s =
   314.1593
p =
   62.8319
```

采用 help 命令或 lookfor 命令可以显示出注释说明部分的内容,其功能和一般 MATLAB 函数的帮助信息是一致的。

利用 help 命令可查询 fcircle 函数的注释说明:

```
help fcircle
```

屏幕显示:

```
CIRCLE     calculate the area and perimeter of a circle of radii r
r          圆半径
s          圆面积
p          圆周长
```

再用 lookfor 命令在第一注释行查询指定的关键词:

```
lookfor perimeter
```

屏幕显示:

```
fcircle.m: %CIRCLE   calculate the area and perimeter of a circle of radii r
```

3.5.2 函数调用

函数文件编制好后,就可调用函数进行计算了。如上面定义 fcircle 函数后,调用它求半径为10的圆的面积和周长。函数调用的一般格式是:

[输出实参表] = 函数名(输入实参表)

要注意的是,函数调用时各实参出现的顺序、个数,应与函数定义时形参的顺序、个数一致,否则会出错。函数调用时,先将实参传递给相应的形参,从而实现参数传递,然后再执行函数的功能。

例 3.15 利用函数文件,实现直角坐标(x,y)与极坐标(ρ,θ)之间的转换。

已知转换公式为:

极坐标的矢径:$\rho=\sqrt{x^2+y^2}$

极坐标的极角：$\theta = \arctan\left(\frac{y}{x}\right)$

函数文件 tran.m 如下：

```
function [rho,theta] = tran(x,y)
rho = sqrt(x * x + y * y);
theta = atan(y/x);
```

调用 tran.m 的命令文件 main1.m 如下：

```
x = input('Please input x = :');
y = input('Please input y = :');
[rho,the] = tran(x,y);
rho
the
```

在 MATLAB 中，函数可以嵌套调用，即一个函数可以调用别的函数，甚至调用它自身(递归调用)。

例 3.16　利用函数的递归调用，求 $n!$。

递归调用函数文件 factor.m 如下：

```
function f = factor(n)
if n < = 1
   f = 1;
else
    f = factor(n - 1) * n;
end
```

在命令文件 main2.m 中调用函数文件 factor.m 求 $s = 1! + 2! + 3! + 4! + 5!$。

```
s = 0;
for i = 1:5
   s = s + factor(i);
end
s
```

3.5.3　函数所传递参数的可调性

MATLAB 在函数调用上有一个与一般高级语言不同之处，就是函数所传递参数数目的可调性。凭借这一点，一个函数可完成多种功能。

在调用函数时，MATLAB 用两个永久变量 *nargin* 和 *nargout* 分别记录调用该函数时的输入实参和输出实参的个数。只要在函数文件中包含这两个变量，就可以准确地知道该函数文件被调用时的输入输出参数个数，从而决定函数如何进行处理。

例 3.17　*nargin* 用法示例。

函数文件 examp.m 如下：

```
function fout = charray(a,b,c)
if nargin = = 1
    fout = a;
```

```
elseif nargin = = 2
      fout = a + b;
elseif nargin = = 3
    fout = (a * b * c)/2;
end
```

命令文件 mydemo.m 如下：

```
x = 1:3;y = [1;2;3];
examp(x)
examp(x,y ')
examp(x,y,3)
```

执行 mydemo.m 后的输出是：

```
ans =
      1     2     3
ans =
      2     4     6
ans =
     21
```

在命令文件 mydemo.m 中，3 次调用函数文件 examp.m，因输入参数的个数分别是 1 个、2 个、3 个，从而执行不同的操作，返回不同的函数值。

3.6 全局变量和局部变量

在 MATLAB 中，全局变量用命令 global 定义。函数文件的内部变量是局部的，与其他函数文件及 MATLAB 工作空间相互隔离。但是，如果在若干函数中，都把某一变量定义为全局变量，那么这些函数将公用这一个变量。全局变量的作用域是整个 MATLAB 工作空间，即全程有效。所有的函数都可以对它进行存取和修改。因此，定义全局变量是函数间传递信息的一种手段。

值得指出，在程序设计中，全局变量固然可以带来某些方便，但却破坏了函数对变量的封装，降低了程序的可读性。因而，在结构化程序设计中，全局变量是不受欢迎的。尤其当程序较大，子程序较多时，全局变量将给程序调试和维护带来不便，故不提倡使用全局变量。如果一定要用全局变量，最好给它起一个能反应变量含义的名字，以免和其他变量混淆。

例 3.18 全局变量应用示例。

先建立函数文件 wadd.m，该函数将输入的参数加权相加。

```
function f = wadd(x,y)
global ALPHA BETA
f = ALPHA * x + BETA * y;
```

再在命令窗口中输入：

```
global ALPHA BETA
ALPHA = 1;
BETA = 2;
```

```
s = wadd(1,2)
```

输出为：

```
s =
    5
```

由于在函数 wadd 和基本工作空间中都把 ALPHA 和 BETA 两个变量定义为全局变量，所以只要在命令窗口中改变 ALPHA 和 BETA 的值，就可改变加权值，而无需修改 wadd.m 文件。

上例只在函数 wadd 和命令窗口中把 ALPHA 和 BETA 变量定义为全局变量，在实际编程时，可在所有需要调用全局变量的函数里定义全局变量，这样就可实现数据共享。为了在基本工作空间中使用全局变量，也要定义全局变量。

在函数文件里，全局变量的定义语句应放在变量使用以前，为了便于了解所有的全局变量，一般把全局变量的定义语句放在文件的前部。

3.7 类和对象

MATLAB 从 5.0 版开始引入了类(Class)和对象(Object)的概念。从结构的概念中我们知道，结构的成员是以变量形式出现的。类的概念是结构的拓展，在类中，不但可以包含变量成员，还可以包含与这些变量相关联的函数或运算。对象是类的一个具体实例。

类和对象的概念是面向对象程序设计(Object－oriented Programming)的基础。采用面向对象技术可以把复杂的操作过程加以隐藏，而外部呈现为人们所习惯的处理形式。在 MATLAB 中，要建立一个类，通常包含以下内容：

(1) 建立类目录。类目录名必须以@开头，@后面紧接待建类的名字。类目录要处在 MATLAB 搜索路径上某目录的子目录位置上。类目录下存放着定义该类对象全部性质和操作方式的一组方法(Methods)文件。

(2) 确定待建类的数据结构。在 MATLAB 中，常用结构矩阵作为类的数据结构。

(3) 对象构造函数。在 MATLAB 中，没有 C++ 中的所谓类声明语句，每个对象都是由构造函数动态产生的。构造函数必须与待建的类同名，并放在相应的类目录上。构造函数的主要内容有：在无输入、本类输入、非本类输入情况下，控制有合理的新对象输出；定义类对象；确定该类的优先等级；对该类是否具有继承性加以定义。

(4) 显示函数。MATLAB 运行时，若一个语句不以分号结束，那么就会在屏幕上自动显示该语句产生的变量。显示变量的具体过程是：MATLAB 首先根据所产生变量的类别，到相应类目录上去调用 display 函数实现屏幕显示。如果找不到相应类目录，则使用内建的显示函数实现屏幕显示。对于新建的类，必须为其设计相应的显示函数。显示函数和前面的构造函数是类目录上必须有的两个 M 函数文件。

(5) 其他函数。这包括与其他类之间的转换函数以及新建类的有关基本运算函数。

例 3.19 建立一个整数类 integer，该类含有一个数据成员 n，代表一个 3 位整数。还具有显示整数各位数字之和、各位数字平方和、各位数字立方和等功能。

操作步骤如下：

(1) 在 MATLAB 5.3 版的 work 目录下建立类目录@integer，即\matlabr11\work\@integer。

(2) 确定 integer 类的数据结构，该类含有一个数据成员，即整数 n。

(3) 建立构造函数@integer\integer.m。

```
function p = integer(v)
if isa(v,'integer')
  p = v;                                    %输入变量是 integer 类
else
  p.n = v;p = class(p,'integer');             %输入变量不是 integer 类
end
```

(4) 为 integer 对象定义显示函数@integer\display.m。

```
function display(p)
disp(['The result is ',num2str(p.n)]);    %integer 对象的显示格式与别的不同
```

(5) 为 integer 对象定义其他运算函数。

求各位数字之和的函数@integer\he.m 如下：

```
function p = he(q)
a = fix(q.n/100);              %输入参数要求 integer 对象
b = rem(fix(q.n/10),10);
c = rem(q.n,10);
p = a + b + c;
p = integer(p);                %输出为 integer 对象
```

求各位数字平方和的函数@integer\pow2.m 如下：

```
function p = pow2(q)
a = fix(q.n/100);
b = rem(fix(q.n/10),10);
c = rem(q.n,10);
p = a * a + b * b + c * c;
p = integer(p);
```

求各位数字立方和的函数@integer\pow3.m 如下：

```
function p = pow3(q)
a = fix(q.n/100);
b = rem(fix(q.n/10),10);
c = rem(q.n,10);
p = a * a * a + b * b * b + c * c * c;
p = integer(p);
```

(6) integer 类使用演示。

建立一个 integer 对象并进行类别检查，命令如下：

```
a = integer(123);                     %建立 integer 对象 a
isa(a,'integer')                      %判断 a 是否 integer 对象
ans =
    1
```

显示 integer 对象命令如下：

```
a                          % 显示 integer 对象 a,注意和一般数据显示格式不同
The result is 123
```

对 integer 对象进行有关运算命令如下：

```
he(integer(816))           % 求各位数字之和
The result is 15
pow3(integer(321))         % 求各位数字立方和
The result is 36
pow2(integer(102))         % 求各位数字平方和,注意和 pow2(102)的区别
The result is 5
pow2(102)                  % MATLAB 标准函数,求 2 的 102 次方
ans =
  5.0706e+030
```

利用 methods 命令列出 integer 类已定义的方法函数名：

```
methods integer
Methods for class integer:

display   he   integer   pow2   pow3
```

3.8　文件操作

文件操作是一种重要的输入输出方式,即从数据文件读取数据或将结果写入数据文件。MATLAB 提供了一系列低层输入输出函数,专门用于文件操作。

3.8.1　文件的打开与关闭

1. 打开文件

在读写文件之前,必须先用 fopen 函数打开文件,并指定允许对该文件进行的操作。fopen 函数的调用格式为：

```
Fid = fopen(文件名,打开方式)
```

其中 *Fid* 用于存储文件句柄值,如果句柄值大于 0,则说明文件打开成功。文件名用字符串形式,表示待打开的数据文件。常见的打开方式有：r(读)、w(写)、a(追加)、r+(可读可写)等。例如,要对数据文件 std.dat 进行读操作,打开该文件：

```
Fid = fopen('std.dat','r')
```

2. 关闭文件

文件在进行完读、写等操作后,应及时关闭,以免数据丢失。关闭文件用 fclose 函数,调用格式为：

```
Sta = fclose(Fid)
```

该函数关闭 *Fid* 所表示的文件。*Sta* 表示关闭文件操作的返回代码,若关闭成功,返回 0,否则返回 −1。如果要关闭所有已打开的文件用 fclose('all')。

3.8.2 二进制文件读写操作

1. 读二进制文件

fread 函数可以读取二进制文件的数据,并将数据存入矩阵。其调用格式为:

```
[A,COUNT] = fread(Fid,size, precision)
```

其中 *A* 用于存放读取的数据,*COUNT* 返回所读取的数据元素个数,*Fid* 为文件句柄,*size* 为可选项,若不选用则读取整个文件内容,若选用则它的值可以是下列值:*N*(读取 *N* 个元素到一个列向量)、inf(读取整个文件)、[M,N](读数据到 $M \times N$ 的矩阵中,数据按列存放)。*precision* 代表数据精度,常用的数据精度有:char、uchar、int、long、float、double 等。缺省数据精度为 uchar,即无符号字符格式。例如

```
Fid = fopen('std.dat', 'r');
A = fread(Fid, 100, 'long');
Sta = fclose(fid);
```

以读方式打开数据文件 std.dat,然后按长整型数据格式读取该文件的前 100 个数据放入向量 ***A***,最后关闭文件。

2. 写二进制文件

fwrite 函数按照指定的数据精度将矩阵中的元素写入到文件中。其调用格式为:

```
COUNT = fwrite (Fid, A, precision)
```

其中 *COUNT* 返回所写的数据元素个数,*Fid* 为文件句柄,***A*** 用来存放写入文件的数据,*precision* 用于控制所写数据的精度,其形式与 fread 函数相同。例如

```
Fid = fopen('aaa.bin', 'w');
fwrite(Fid, X, 'float');
```

将矩阵 ***X*** 中的数据用浮点格式写入 aaa.bin 文件。

3.8.3 文本文件读写操作

1. 读文本文件

fscanf 函数可以读取文本文件的内容,并按指定格式存入矩阵。其调用格式为:

```
[A,COUNT] = fscanf(Fid,format,size)
```

其中 ***A*** 用以存放读取的数据,*COUNT* 返回所读取的数据元素个数。*Fid* 为文件句柄。*format* 用以控制读取的数据格式,由%加上格式符组成,常见的格式符有:d(整型)、f(浮点型)、s(字符串型)、c(字符型)等,在%与格式符之间还可以插入附加格式说明符,如数据宽度说明等。*size* 为可选项,决定矩阵 ***A*** 中数据的排列形式,它可以取下列值:*N*(读取 *N* 个元素到一个列向量)、inf(读取整个文件)、[M,N](读数据到 $M \times N$ 的矩阵中,数据按列存放)。例如,从指定文件中读取 100 个整数,并存入向量 ***x*** 中:

```
x = fscanf(Fid, '%5d',100);
```

也可以将读取的 100 个整数存入 10×10 矩阵 ***y*** 中:

```
y = fscanf(Fid, '%5d',[10,10]);
```

2. 写文本文件

fprintf 函数可以将数据按指定格式写入到文本文件中。其调用格式为:

```
COUNT = fprintf(Fid, format, A)
```

其中 *A* 存放要写入文件的数据。先按 *format* 指定的格式将数据矩阵 *A* 格式化，然后写入到 *Fid* 所指定的文件。格式符与 fscanf 函数相同。例如

```
x = 0: 0.1: 1;
y = [x; exp(x)];
Fid = fopen('exp.txt', 'w');
fprintf(Fid,'%6.2f   %12.8f\n',y);
fclose(Fid);
```

上述程序段建立一文本文件 exp.txt，并将矩阵 *y* 的列向量数据分别以浮点格式%6.2f 和%12.8f写入文本文件 exp.txt。格式串中的 \ *n* 表示换行。由于是文本文件，可以在 MATLAB 命令窗口用 type 命令显示其内容。

例 3.20 从键盘输入5名学生的姓名和成绩，先建立一个数据文件，然后读出数据文件的内容，并按成绩从低到高的顺序在屏幕上输出学生信息。

程序如下：

```
fid = fopen('t.txt','w');
for i = 1:5
    n = input('name = ? ','s');
    s = input('score = ? ');
    fprintf(fid,'%8s%6.1f\n',n,s);
end
fclose(fid);
fid = fopen('t.txt','r');
n = 1;
while (1)
    stu(n).nam = fscanf(fid,'%s',1);
    stu(n).sco = fscanf(fid,'%f',1);
    if feof(fid) = = 1        %feof 函数测试是否到了文件结束位置
        break;
    end
    score(n) = stu(n).sco;
    n = n + 1;
end
n = n - 1;
[sortsco,index] = sort(score);      %sort 函数将向量排序
for i = 1:n
    oup = ['姓名',stu(index(i)).nam,'   成绩',num2str(sortsco(i))];
    disp(oup);
end
fclose(fid);
```

3.8.4 数据文件定位

MATLAB 提供了与文件定位操作有关的函数 fseek 和 ftell。通过这两个函数，用户可以设定

或获取文件指针位置，以方便、灵活地进行输入输出操作。

fseek 函数用于定位文件位置指针，其调用格式为：

```
status = fseek(Fid, offset, origin)
```

其中 *Fid* 为文件句柄，*offset* 表示位置指针相对移动的字节数，若为正整数表示向文件尾方向移动，若为负整数表示向文件头方向移动，*origin* 表示位置指针移动的参照位置，它的取值有 3 种可能：cof 表示文件的当前位置，bof 表示文件的开始位置，'eof'表示文件的结束位置。若定位成功，*status* 返回值为 0，否则返回值为 -1。

ftell 函数返回文件指针的当前位置，其调用格式为：

```
position = ftell (Fid)
```

返回值为从文件开始到指针当前位置的字节数。若返回值为 -1 表示获取文件当前位置失败。

例 3.21 分析下列程序执行后，*four*、*position* 和 *three* 3 个变量的值。

```
a = 1:5;
Fid = fopen('fdat.bin', 'w');           %以写方式打开文件 fdat.bin
fwrite(Fid, a, 'int16');                %将 a 的元素以双字节整型写入文件 fdat.bin
status = fclose(Fid);
Fid = fopen('fdat.bin', 'r');           %以读方式打开文件 fdat.bin
status = fseek(Fid, 6, 'bof');          %将文件指针从开始位置向尾部移动 6 个字节
four = fread(Fid, 1, 'int16');          %读取第 4 个数据,并移动指针到下一个数据
position = ftell(Fid);                  %ftell 的返回值为 8
status = fseek(Fid, -4, 'cof');         %将文件指针从当前位置往前移动 4 个字节
three = fread(Fid, 1, 'int16');         %读取第 3 个数据
status = fclose(Fid);
```

程序的分析过程见注释。程序执行后，*four*、*position* 和 *three* 的值分别是 4,8 和 3。

习 题 三

1. 从键盘输入一个 4 位整数，按如下规则加密后输出。加密规则：每位数字都加上 7，然后用和除以 10 的余数取代该数字；再把第一位与第三位交换，第二位与第四位交换。

2. 分别用 if 语句和 switch 语句实现以下计算，其中 a,b,c 的值从键盘输入。

$$y=\begin{cases} ax^2+bx+c\ , & 0.5\leqslant x<1.5 \\ a\sin^c b+x\ , & 1.5\leqslant x<3.5 \\ \ln\left|b+\dfrac{c}{x}\right| & 3.5\leqslant x<5.5 \end{cases}$$

3. 输入 20 个数，求其中最大数和最小数。要求分别用循环结构和调用 MATLAB 的 max 函数、min 函数来实现。

4. 已知

$$s=1+2+2^2+2^3+\cdots+2^{63}$$

分别用循环结构和调用 MATLAB 的 sum 函数求 s 的值。

5. 编写一个函数文件，用于求两个矩阵的乘积和点乘，然后在命令文件中调用该函数。

6. 编写一个函数文件，求小于任意自然数 n 的斐波那契(Fibnacci)数列各项。Fibnacci 数列定义如下：

$$\begin{cases} f_1 = 1 \\ f_2 = 1 \\ f_n = f_{n-1} + f_{n-2}, \quad n > 2 \end{cases}$$

7. 产生 20 个两位随机整数，输出其中小于平均值的偶数。

8. 试定义一个字符串类 string，使其至少具有内容(contents)和长度(length)两个数据成员，并具有显示字符串、求字符串长度、在原字符串后添加一个字符串等功能。

9. 编写程序，该程序能读取一个文本文件，并能将文本文件中的小写字母转换为相应的大写字母而生成一个新的文本文件。

10. 写出下列程序的输出结果。

(1)
```
s = 0;
a = [12,13,14;15,16,17;18,19,20;21,22,23];
for k = a
    for j = 1:4
        if rem(k(j),2) ~ = 0
            s = s + k(j);
        end
    end
end
s
```

(2)
```
N = input(' N = ');
c(1:2 * N - 1) = ' * ';
for i = 1:N
    c1(1:16) = '';
    k = N + 1 - i;
    c1(k:k + 2 * i - 2) = c(1:2 * i - 1);
    disp(c1);
end
```
当 N 的值取 5 时，写出输出结果。

(3) 命令文件 ex82.m：
```
global x
x = 1:2:5;y = 2:2:6;
exsub(y);
x
y
```
函数文件 exsub.m：
```
function fun = sub(z)
global x
z = 3 * x; x = x + z;
```

(4) 函数文件 mult.m：
```
function a = mult(var)
a = var{1};
for i = 2:length(var)
    a = a * var{i};
end
```
命令文件 pp.m：
```
p = [17, -6;35, -12];
p = mult({p;p;p;p;p})
```

第4章 MATLAB绘图

强大的绘图功能是MATLAB的特点之一。MATLAB提供了一系列的绘图函数,用户不需过多考虑绘图细节,只需给出一些基本参数就能得到所需图形,这一类函数称为高层绘图函数。除此之外,MATLAB还提供了直接对图形句柄进行操作的低层绘图操作。这类操作将图形的每个图形元素(如坐标轴、曲线、曲面或文字等)看作是一个独立的对象,系统给每个图形对象分配一个句柄,以后可以通过该句柄对该图形元素进行操作,而不影响图形的其他部分。高层绘图操作简单明了、方便高效,是用户最常使用的绘图方法,而低层绘图操作控制和表现图形的能力更强,为用户更加自主地绘制图形创造了条件。事实上,MATLAB的高层绘图函数都是利用低层绘图函数而建立起来的。

本章介绍绘制二维和三维图形的高层绘图函数以及其他图形控制函数的使用方法,在此基础上,再介绍可以操作和控制各种图形对象的低层绘图操作。

4.1 二维图形

4.1.1 绘制二维曲线的最基本函数

在MATLAB中,最基本且应用最为广泛的绘图函数为plot函数,利用它可以在二维平面上绘制出不同的曲线。

1. plot函数的基本用法

plot函数用于绘制 $x-y$ 平面上的线性坐标曲线图,因此需提供一组 x 坐标以及与各点 x 坐标对应的 y 坐标,这样就可以绘制分别以 x 坐标和 y 坐标为横、纵坐标的二维曲线。plot函数的基本调用格式为:

```
plot(x,y)
```

其中 x 和 y 为长度相同的向量,分别用于存储 x 坐标和 y 坐标数据。

例4.1 在 $0\leqslant x\leqslant 2\pi$ 区间内,绘制曲线 $y=2e^{-0.5x}\sin(2\pi x)$。

程序如下:

```
x = 0:pi/100:2 * pi;
y = 2 * exp( - 0.5 * x) . * sin(2 * pi * x);
plot(x,y)
```

程序执行后,打开一个图形窗口,在其中绘出图4.1所示的曲线。

注意:求 y 时,指数函数和正弦函数之间要用点乘运算,而因2为标量,所以2与指数函数之间可以用乘法运算。这样,x 和 y 所包含的元素个数相等,$y(i)$ 是 $x(i)$ 点的函数值。

以上提到plot函数的自变量 x,y 为长度相同的向量,这是最常见、最基本的情况。实际应

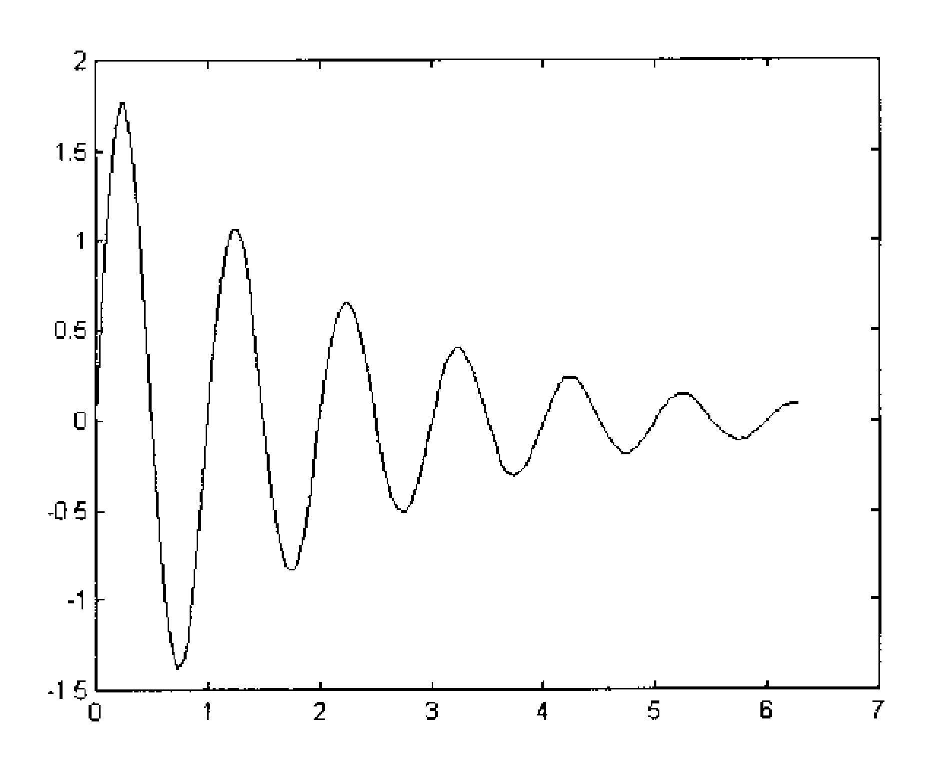


图 4.1　$y = 2e^{0.5x}\sin(2\pi x)$的曲线

用中还有一些变化，下面分别说明。

(1) 当 x，y 是同维矩阵时，则以 x，y 对应列元素为横、纵坐标分别绘制曲线，曲线条数等于矩阵的列数。

(2) 当 x 是向量，y 是有一维与 x 同维的矩阵时，则绘制出多根不同色彩的曲线。曲线条数等于 y 矩阵的另一维数，x 被作为这些曲线共同的横坐标。例如，下列程序可以在同一坐标中同时绘制出正弦和余弦曲线。

```
x = linspace(0,2 * pi,100);
y = [sin(x);cos(x)];
plot(x,y)
```

程序首先产生一个行向量 x，然后分别求取行向量 sin(x)和 cos(x)并将它们构成矩阵 y 的两行，最后在同一坐标中同时绘制出两条曲线。

(3) plot 函数最简单的调用格式是只包含一个输入参数：plot(x)。在这种情况下，当 x 是实向量时，以该向量元素的下标为横坐标，元素值为纵坐标画出一条连续曲线，这实际上是绘制折线图。当 x 是实矩阵时，则按列绘制每列元素值相对其下标的曲线，曲线条数等于 x 阵的列数。当 x 是复数矩阵时，则按列分别以元素实部和虚部为横、纵坐标绘制多条曲线。

例 4.2　某工厂2000年各月总产值(单位：万元)分别为 22、60、88、95、56、23、9、10、14、81、56、23，试绘制折线图以显示出该厂总产值的变化情况。

程序如下：

```
p = [22,60,88,95,56,23,9,10,14,81,56,23];
plot(p)
```

结果如图 4.2 所示。

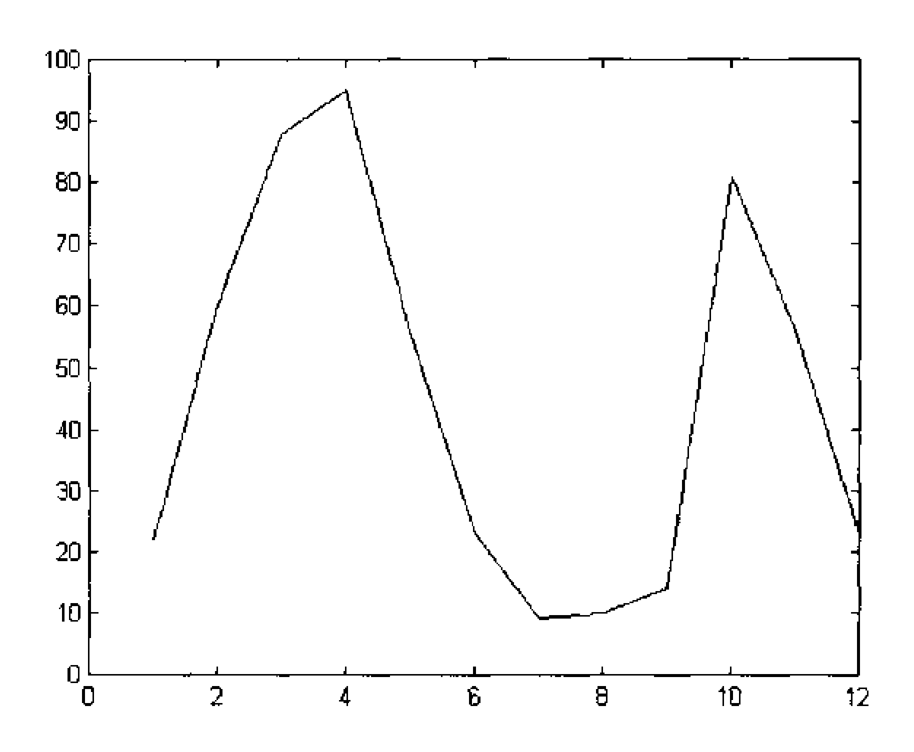


图 4.2 某工厂 2000 年产值变动曲线

2. 含多个输入参数的 plot 函数

plot 函数可以包含若干组向量对，每一向量对可以绘制出一条曲线。含多个输入参数的 plot 函数调用格式为：

```
plot(x1,y1,x2,y2,…,xn,yn)
```

其中 $x1$ 和 $y1$，$x2$ 和 $y2$，…，xn 和 yn 分别组成一组向量对，每一组向量对的长度可以不同。每一向量对可以绘制出一条曲线，这样可以在同一坐标内绘制出多条曲线。

3. 含选项的 plot 函数

MATLAB 提供了一些绘图选项，用于确定所绘曲线的线型、颜色和数据点标记符号。这些选项如表 4.1 所示，它们可以组合使用。例如，b-.表示蓝色点划线，y:d 表示黄色虚线并用菱形符标记数据点。当选项省略时，MATLAB 规定，线型一律用实线，颜色将根据曲线的先后顺序依次采用表 4.1 给出的前 7 种颜色。含选项的 plot 函数调用格式为：

```
plot(x1,y1,选项 1,x2,y2,选项 2,…,xn,yn,选项 n)
```

例 4.3 用不同线型和颜色在同一坐标内绘制曲线 $y=2e^{-0.5x}\sin(2\pi x)$ 及其包络线。

程序如下：

```
x=(0:pi/100:2*pi)';
y1=2*exp(-0.5*x)*[1,-1];
y2=2*exp(-0.5*x).*sin(2*pi*x);
x1=(0:12)/2;
y3=2*exp(-0.5*x1).*sin(2*pi*x1);
plot(x,y1,'g:',x,y2,'b--',x1,y3,'rp');
```

表 4.1　线型、颜色和标记符号选项

(a) 线型选项

线型	说明
-	实线
:	虚线
-.	点划线
--	双划线

(b) 颜色选项

颜色	说明
b	蓝色
g	绿色
r	红色
c	青色
m	品红色
y	黄色
k	黑色
w	白色

(c) 标记符号选项

标记符号	说明	标记符号	说明
.	点	s	方块符(square)
o	圆圈	d	菱形符(diamond)
x	叉号	v	朝下三角符号
+	加号	^	朝上三角符号
*	星号	<	朝左三角符号
		>	朝右三角符号
		p	五角星符(pentagram)
		h	六角星符(hexagram)

程序运行结果如图 4.3 所示。plot 函数中包含 3 组绘图参数,第一组用绿色虚线绘出两根包络线,第二组用蓝色双划线绘出曲线 y,第三组用红色五角星离散地标出数据点。

程序中第一条命令用矩阵转置运算符将行向量转换为列向量。请读者思考一下,如果这里不用转置操作,程序会怎样?如果要得到图 4.3 的图形,程序应如何修改?

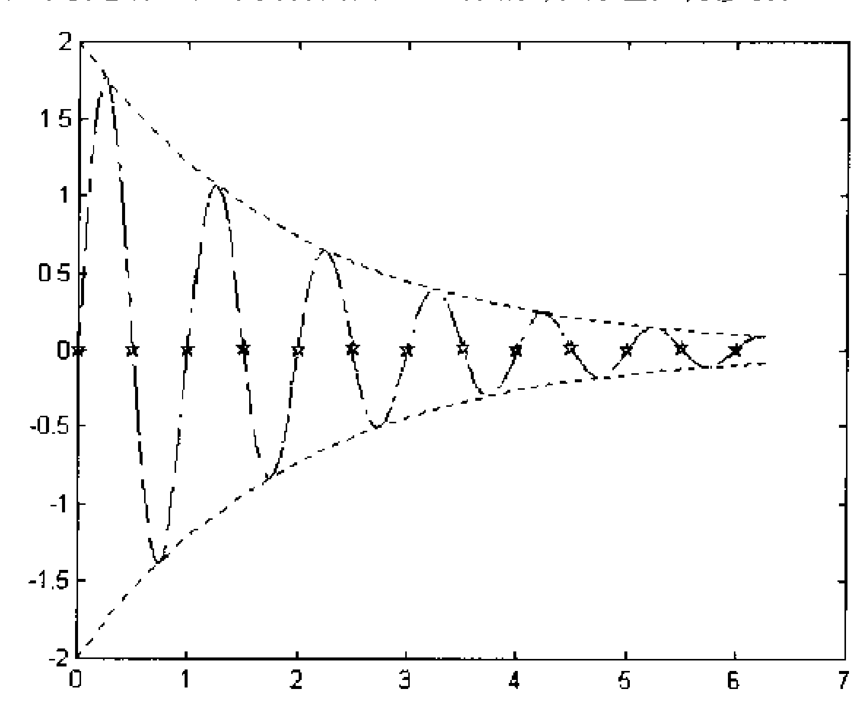


图 4.3　用不同线型和颜色绘制的曲线

4. 双纵坐标函数 plotyy

plotyy 函数是 MATLAB 5.x 新增的函数。它能把函数值具有不同量纲、不同数量级的两个函数绘制在同一坐标中。调用格式为:

```
plotyy(x1,y1,x2,y2)
```

其中 $x1-y1$ 对应一条曲线,$x2-y2$ 对应另一条曲线。横坐标的标度相同,纵坐标有 2 个,左纵坐标用于 $x1-y1$ 数据对,右纵坐标用于 $x2-y2$ 数据对。

例 4.4　用不同标度在同一坐标内绘制曲线 $y1=e^{-0.5x}\sin(2\pi x)$ 及曲线 $y2=1.5e^{-0.1x}\cdot\sin(x)$。

程序如下：

```
x1 = 0:pi/100:2 * pi;
x2 = 0:pi/100:3 * pi;
y1 = exp( - 0.5 * x1) . * sin(2 * pi * x1);
y2 = 1.5 * exp( - 0.1 * x2) . * sin(x2);
plotyy(x1,y1,x2,y2);
```

程序运行结果如图 4.4 所示。

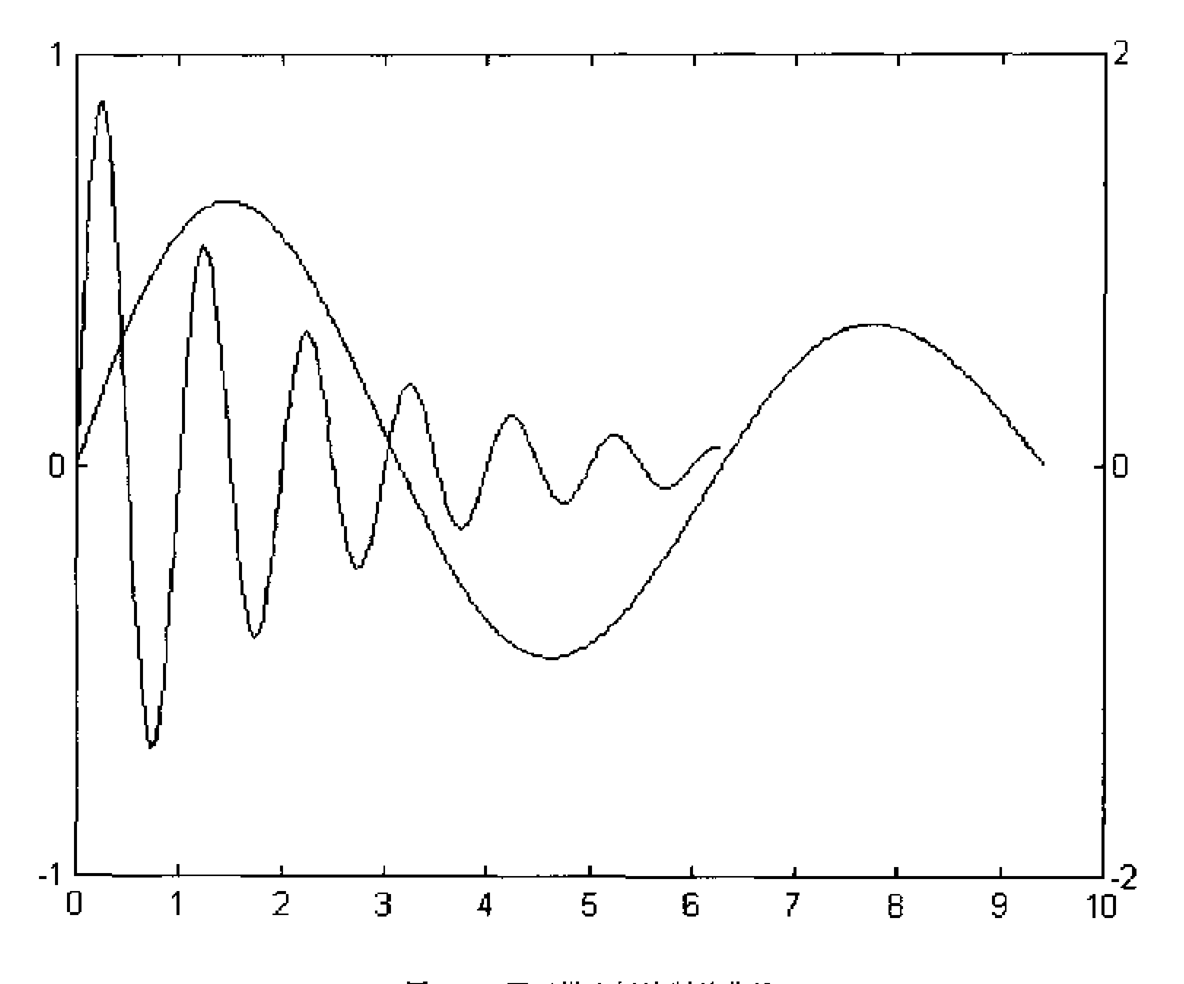


图 4.4　用双纵坐标绘制的曲线

4.1.2　绘制图形的辅助操作

绘制完图形后，可能还需要对图形进行一些辅助操作，以使图形意义更加明确，可读性更强。

1. 图形标注

在绘制图形的同时，可以对图形加上一些说明，如图形名称、坐标轴说明以及图形某一部分的含义等，这些操作称为添加图形标注。有关图形标注函数的调用格式为：

```
title(图形名称)
xlabel(x 轴说明)
ylabel(y 轴说明)
text(x,y,图形说明)
legend(图例 1,图例 2,…)
```

其中 title 和 xlabel、ylabel 函数分别用于说明图形和坐标轴的名称。text 函数是在(x,y)坐标处添加图形说明。添加文本说明也可用 gtext 命令,执行该命令时,十字坐标光标自动跟随鼠标移动,单击鼠标即可将文本放置在十字光标处。如命令 gtext('cos(x)'),即可放置字符串'cos(x)'。legend 函数用于绘制曲线所用线型、颜色或数据点标记图例,图例放置在图形空白处,用户还可以通过鼠标移动图例,将其放到所希望的位置。

应当说明的是,除 legend 函数外,其他函数同样适用于三维图形,z 坐标轴说明用 zlabel 函数。

例 4.5 给图4.3中的图形添加图形标注。

程序如下:

```
x = (0:pi/100:2 * pi)';
y1 = 2 * exp( - 0.5 * x) * [1, - 1];
y2 = 2 * exp( - 0.5 * x) . * sin(2 * pi * x);
x1 = (0:12)/2;
y3 = 2 * exp( - 0.5 * x1) . * sin(2 * pi * x1);
plot(x,y1,'g:',x,y2,'b--',x1,y3,'rp');
title('曲线及其包络线');                    %加图形标题
xlabel('independent variable X');          %加 X 轴说明
ylabel('independent variable Y');          %加 Y 轴说明
text(2.8,0.5,'包络线');                     %在指定位置添加图形说明
text(0.5,0.5,'曲线 y');
text(1.4,0.1,'离散数据点');
legend('包络线','包络线','曲线 y','离散数据点') %加图例
```

程序运行结果如图 4.5 所示。

2. 坐标控制

在绘制图形时,MATLAB 可以自动根据要绘制曲线数据的范围选择合适的坐标刻度,使得曲线能够尽可能清晰地显示出来。所以,在一般情况下用户不必选择坐标轴的刻度范围。但是,如果用户对坐标系不满意,可利用 axis 函数对其重新设定。该函数的调用格式为:

```
axis([xmin,xmax,ymin,ymax,zmin,zmax])
```

如果只给出前 4 个参数,则 MATLAB 按照给出的 x, y 轴的最小值和最大值选择坐标系范围,以便绘制出合适的二维曲线。如果给出了全部参数,则系统按照给出的 3 个坐标轴的最小值和最大值选择坐标系范围,以便绘制出合适的三维图形。

axis 函数功能丰富,常用的用法还有:

axis equal　　纵、横坐标轴采用等长刻度

axis square　　产生正方形坐标系(缺省为矩形)

axis auto　　使用缺省设置

axis off　　取消坐标轴

axis on　　显示坐标轴

一般情况下,绘图命令每执行一次就刷新当前图形窗口,图形窗口原有图形将不复存在。若希望在已存在的图形上再继续添加新的图形,可使用图形保持命令 hold。hold on/off 命令控制是

保持原有图形还是刷新原有图形，不带参数的 hold 命令在两种状态之间进行切换。

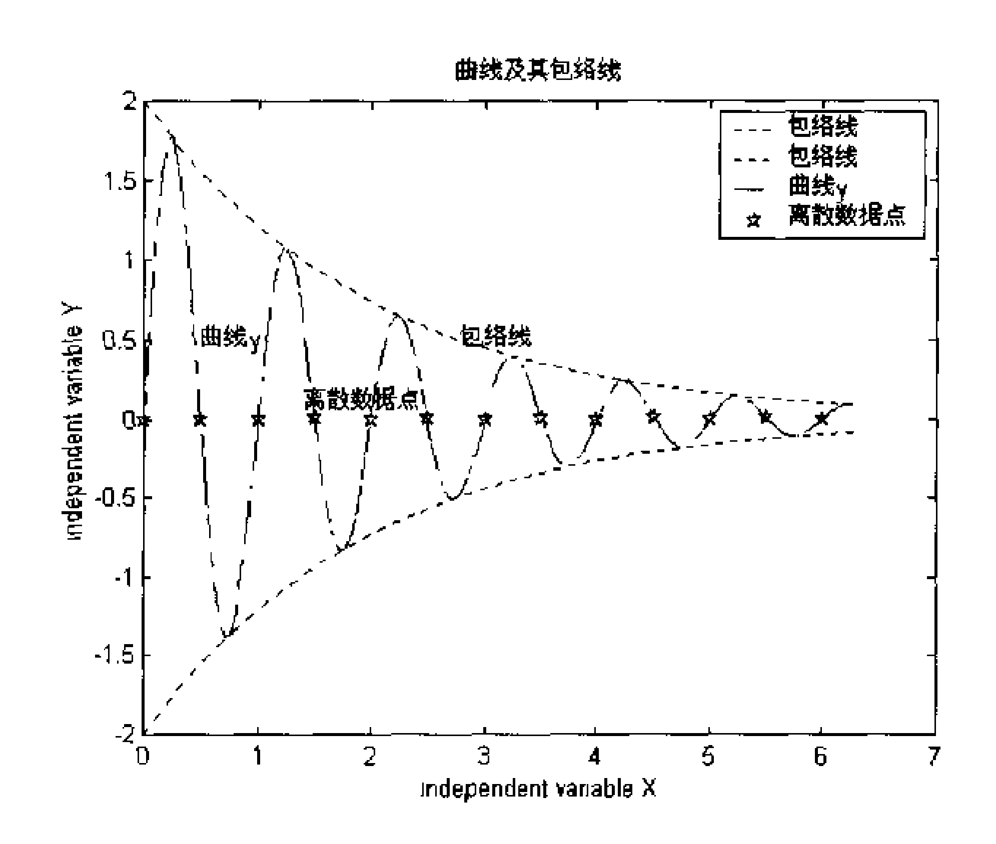


图 4.5 给图形加图形标注

给坐标加网格线用 grid 命令来控制。grid on/off 命令控制是画还是不画网格线，不带参数的 grid 命令在两种状态之间进行切换。

给坐标加边框用 box 命令。box on/off 命令控制是加还是不加边框线，不带参数的 box 命令在两种状态之间进行切换。

例 4.6 用图形保持功能在同一坐标内绘制曲线 $y=2e^{-0.5x}\sin(2\pi x)$ 及其包络线，并加网格线。

程序如下：

```
x = (0:pi/100:2 * pi)';
y1 = 2 * exp( - 0.5 * x) * [1, - 1]; y2 = 2 * exp( - 0.5 * x) . * sin(2 * pi * x);
plot(x,y1,' b:');
axis([0,2 * pi, - 2,2]);          %设置坐标
hold on;                          %设置图形保持状态
plot(x,y2,' k ');
grid on;                          %加网格线
box off;                          %不加坐标边框
hold off;                         %关闭图形保持
```

程序运行结果如图 4.6 所示。

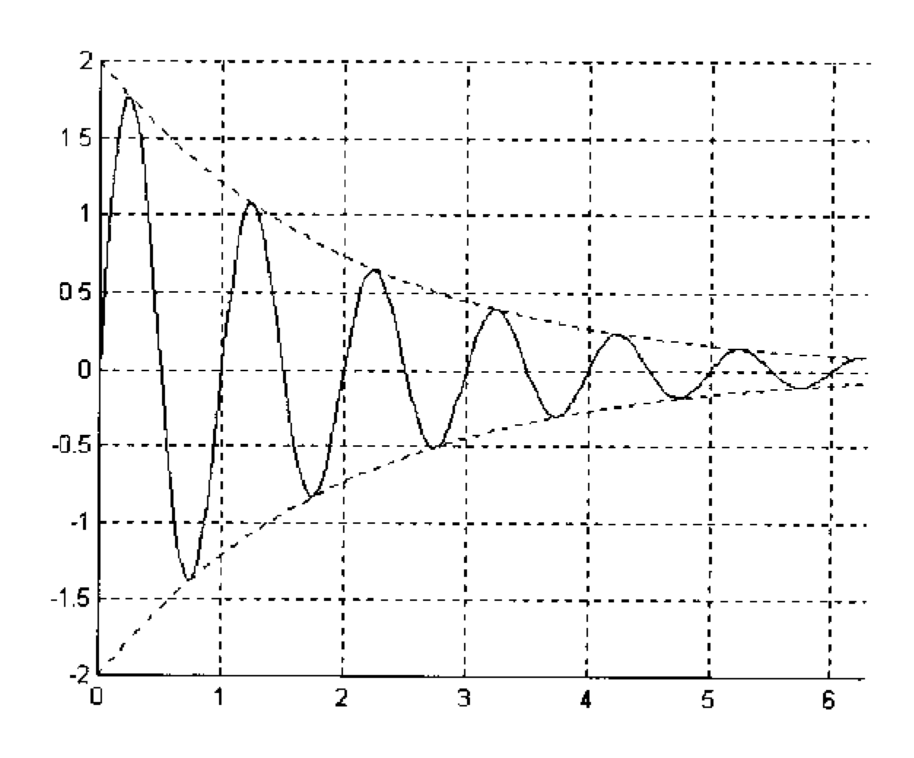


图 4.6　坐标控制

3．图形窗口的分割

在实际应用中，经常需要在一个图形窗口内绘制若干个独立的图形，这就需要对图形窗口进行分割。分割后的图形窗口由若干个绘图区组成，每一个绘图区可以建立独立的坐标系并绘制图形。同一图形窗口中的不同图形称为子图。MATLAB 系统提供了 subplot 函数，用来将当前图形窗口分割成若干个绘图区。每个区域代表一个独立的子图，也是一个独立的坐标系，可以通过 subplot 函数激活某一区，该区为活动区，所发出的绘图命令都是作用于活动区域。subplot 函数的调用格式为：

```
subplot(m,n,p)
```

该函数将当前图形窗口分成 $m \times n$ 个绘图区，即每行 n 个，共 m 行，区号按行优先编号，且选定第 p 个区为当前活动区。在每一个绘图区允许以不同的坐标系单独绘制图形。

例 4.7　在一个图形窗口中以子图形式同时绘制正弦、余弦、正切、余切曲线。

程序如下：

```
x = linspace(0,2 * pi,60);
y = sin(x);z = cos(x);
t = sin(x)./(cos(x) + eps); ct = cos(x)./(sin(x) + eps);
subplot(2,2,1);
plot(x,y);title(' sin(x)');axis ([0,2 * pi, - 1,1]);
subplot(2,2,2);
```

```
plot(x,z);title(' cos(x)');axis ([0,2 * pi, - 1,1]);
subplot(2,2,3);
plot(x,t);title(' tangent(x)');axis ([0,2 * pi, - 40,40]);
subplot(2,2,4);
plot(x,ct);title(' cotangent(x)');axis ([0,2 * pi, - 40,40]);
```

程序运行结果如图 4.7 所示。

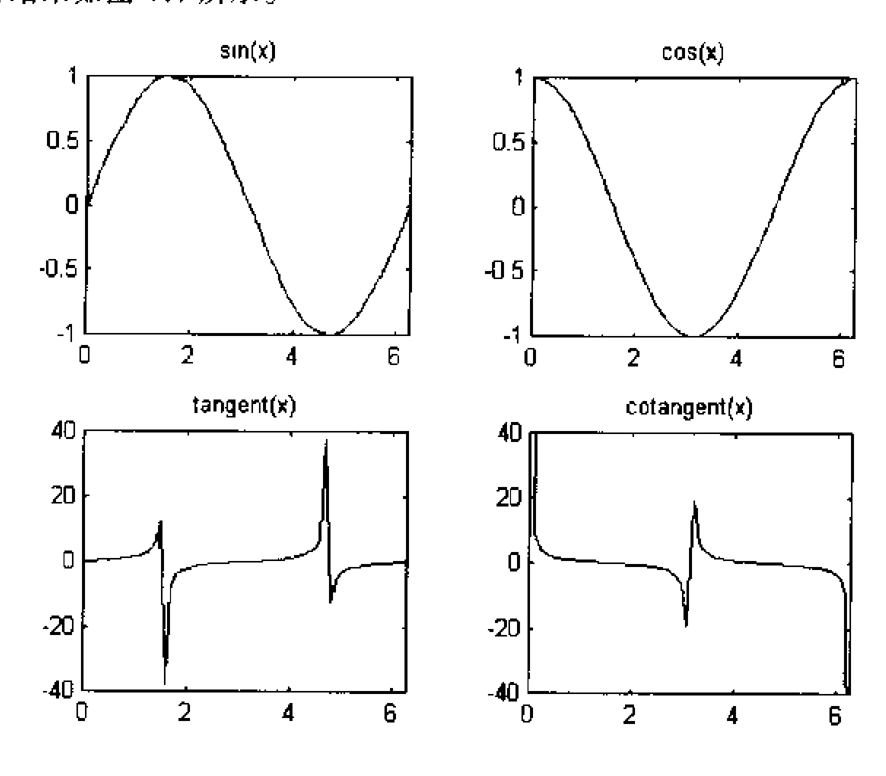


图 4.7 图形窗口的分割

例 4.7 中将图形窗口分割成 2×2 个绘图区,编号从 1 到 4,各区分别绘制一幅图形,这是最规则的情况。实际上,还可以作更灵活的分割。看下面的程序。

```
x = linspace(0,2 * pi,60);
y = sin(x);z = cos(x);
t = sin(x)./(cos(x) + eps);ct = cos(x)./(sin(x) + eps);
subplot(2,2,1);                        %选择 2×2 个区中的 1 号区
stairs(x,y);axis([0,2 * pi, - 1,1]);   %stairs 函数绘制阶梯图(见下一小节)
subplot(2,1,2);                        %选择 2×1 个区中的 2 号区
stem(x,y);axis([0,2 * pi, - 1,1]);     %stem 函数绘制杆图(见下一小节)
subplot(4,4,3);                        %选择 4×4 个区中的 3 号区
plot (x,y);axis([0,2 * pi, - 1,1]);
subplot(4,4,4);                        %选择 4×4 个区中的 4 号区
plot(x,z);axis([0,2 * pi, - 1,1]);
subplot(4,4,7);                        %选择 4×4 个区中的 7 号区
plot(x,t);axis([0,2 * pi, - 40,40]);
subplot(4,4,8);                        %选择 4×4 个区中的 8 号区
```

```
plot(x,ct);axis([0,2*pi,-40,40]);
```

程序运行结果如图 4.8 所示。本质上讲,这还是一种规则的分割,4.5 节将进一步说明,如何利用坐标轴对象操作对图形窗口作任意分割。

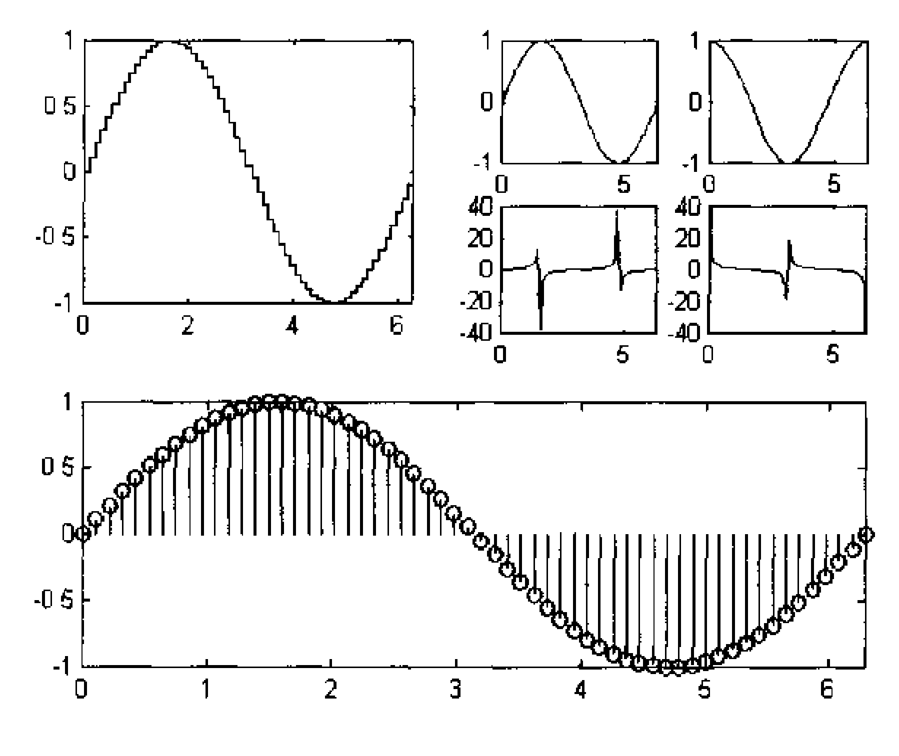


图 4.8　图形窗口的灵活分割

4.1.3　绘制二维图形的其他函数

1. 其他形式的线性直角坐标图

在线性直角坐标系中,其他形式的图形有条形图、阶梯图、杆图和填充图等,所采用的函数分别是:

```
bar(x,y,选项)
stairs(x,y,选项)
stem(x,y,选项)
fill(x1,y1,选项 1,x2,y2,选项 2,…)
```

前 3 个函数的用法与 plot 函数相似,只是没有多输入变量形式。fill 函数按向量元素下标渐增次序依次用直线段连接 x,y 对应元素定义的数据点。假若这样连接所得折线不封闭,那么 MATLAB 将自动把该折线的首尾连接起来,构成封闭多边形,然后将多边形内部涂满指定的颜色。

例 4.8　分别以条形图、填充图、阶梯图和杆图形式绘制曲线 $y=2e^{-0.5x}$。

程序如下:

```
x=0:0.35:7;
y=2*exp(-0.5*x);
subplot(2,2,1);bar(x,y,'g');
title('bar(x,y,''g'')');axis([0,7,0,2]);
```

```
subplot(2,2,2);fill(x,y,'r');
title('fill(x,y,"r")');axis([0,7,0,2]);
subplot(2,2,3);stairs(x,y,'b');
title('stairs(x,y,"b")');axis([0,7,0,2]);
subplot(2,2,4);stem(x,y,'k');
title('stem(x,y,"k")');axis([0,7,0,2]);
```

程序执行结果如图 4.9 所示。

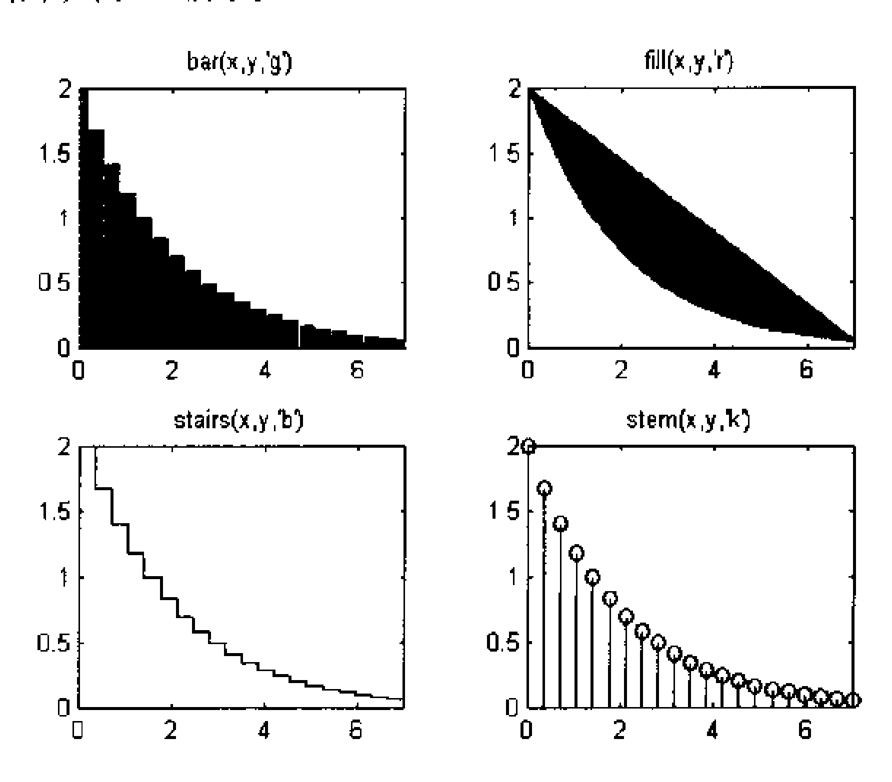


图 4.9　几种不同形式的图形

2. 极坐标图

polar 函数用来绘制极坐标图，其调用格式为：

```
polar(theta,rho,选项)
```

其中 *theta* 为极坐标极角，*rho* 为极坐标矢径，选项的内容与 plot 函数相似。

例 4.9　绘制 $\rho=\sin(2\theta)\cos(2\theta)$ 的极坐标图。

程序如下：

```
theta = 0:0.01:2 * pi;
rho = sin(2 * theta) .* cos(2 * theta);
polar(theta,rho,'k');
```

程序运行结果如图 4.10 所示。

3. 对数坐标图形

在实际应用中，经常用到对数坐标，例如控制理论中的 Bode 图。MATLAB 提供了绘制对数和半对数坐标曲线的函数，调用格式为：

```
semilogx(x1,y1,选项 1,x2,y2,选项 2,…)
```

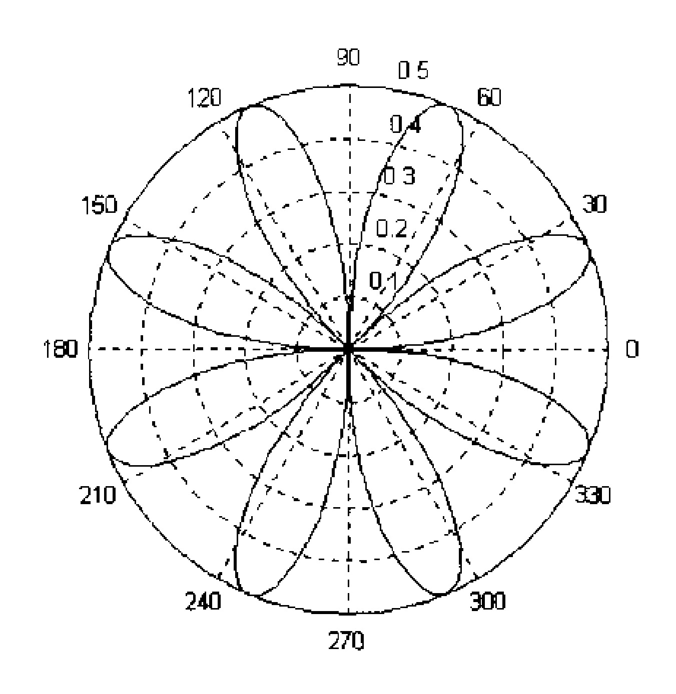


图 4.10　极坐标图

```
semilogy(x1,y1,选项 1,x2,y2,选项 2,…)
loglog(x1,y1,选项 1,x2,y2,选项 2,…)
```

其中选项的定义与 plot 函数完全一致,所不同的是坐标轴的选取。semilogx 函数使用半对数坐标,x 轴为常用对数刻度,而 y 轴仍保持线性刻度。semilogy 函数也使用半对数坐标,y 轴为常用对数刻度,而 x 轴仍保持线性刻度。loglog 函数使用全对数坐标,x,y 轴均采用常用对数刻度。

例 4.10　绘制 $y=10x^2$ 的对数坐标图并与直角线性坐标图进行比较。

程序如下:

```
x = 0:0.1:10;
y = 10 * x. * x;
subplot(2,2,1);plot(x,y);title(' plot(x,y)');grid on;
subplot(2,2,2);semilogx(x,y);title(' semilogx(x,y)');grid on;
subplot(2,2,3);semilogy(x,y);title(' semilogy(x,y)');grid on;
subplot(2,2,4);loglog(x,y);title(' loglog(x,y)');grid on;
```

程序执行结果如图 4.11 所示。

第 2 章曾介绍过利用冒号表达式或 linspace 函数产生线性坐标向量,MATLAB 还提供了一个实用的函数 logspace,它可以按对数等间距地分布来产生一个向量。该函数的调用格式为:

```
logspace(a,b,n )
```

其中 a 和 b 是生成向量的第一个和最后一个元素,n 是元素总数。当 n 省略时,自动产生 50 个元素。

4. 对函数自适应采样的绘图函数

前面介绍了很多绘图函数,基本的操作方法为:先取足够稠密的自变量向量 x,然后计算出函数值向量 y,最后用绘图函数绘图。在取数据点时一般都是等间隔采样,这对绘制高频率变化

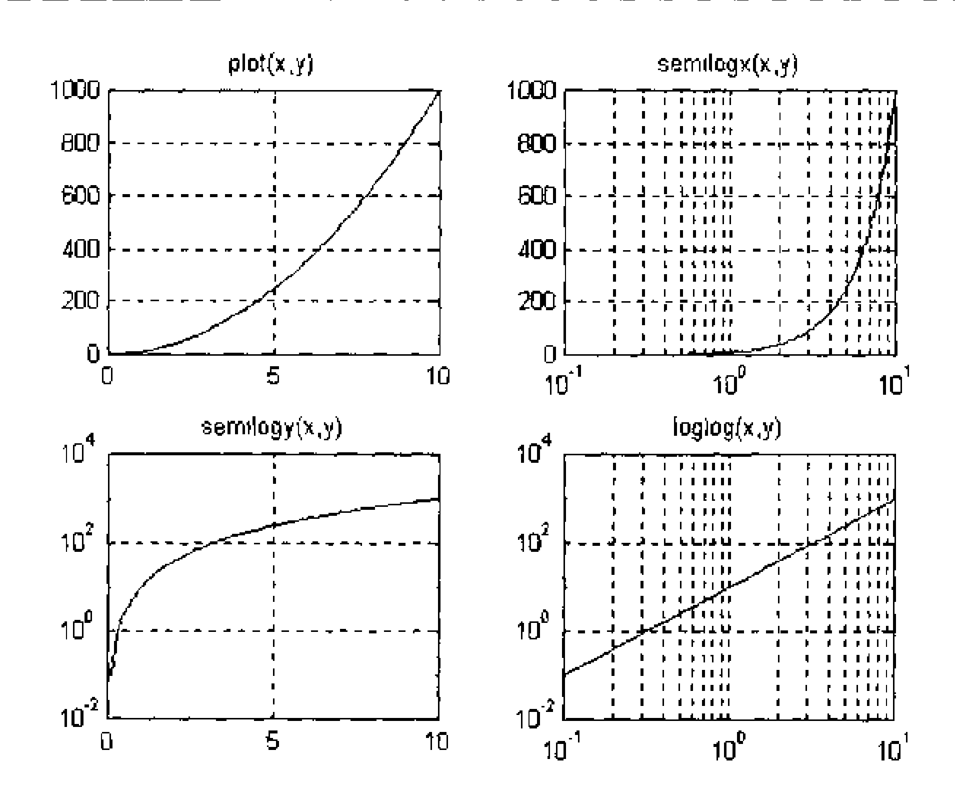


图 4.11　对数坐标图

的函数不够精确。例如函数$f(x)=\cos(\tan(\pi x))$，在(0,1)范围有无限多个振荡周期，函数变化率大。为提高精度，绘制出比较真实的函数曲线，就不能等间隔采样，而必须在变化率大的区段密集采样，以充分反应函数的实际变化规律，进而提高图形的真实度。fplot 函数可自适应地对函数进行采样，能更好地反应函数的变化规律。fplot 函数的调用格式为：

```
fplot(fname,lims,tol,选项)
```

其中 fname 为函数名，以字符串形式出现。它可以是由多个分量函数构成的行向量，分量函数可以是函数的直接字符串，也可以是内部函数名或函数文件名，但自变量都必须为 x。lims 为 x，y 的取值范围，以行向量形式出现，取二元向量[xmin,xmax]时，x 轴的范围被人为确定，取四元向量[xmin,xmax,ymin,ymax]时，x，y 轴的范围被人为确定。*tol* 为相对允许误差，其系统默认值为 2e－3。选项定义与 plot 函数相同。例如

```
fplot('sin(x)',[0,2*pi],'*')
fplot('[sin(x),cos(x)]',[0,2*pi,-1.5,1.5],1e-3,'r.')
```

第二个语句同时绘制正弦、余弦曲线。观察上述语句绘制的正余弦曲线采样点的分布，可发现曲线变化率大的区段，采样点比较密集。

例 4.11　用 fplot 函数绘制 $f(x)=\cos(\tan(\pi x))$的曲线。

先建立函数文件 myf.m，程序如下：

```
function y = myf(x)
  y = cos(tan(pi * x));
```

再用 fplot 函数绘制 myf.m 函数的曲线，命令如下：

```
fplot('myf',[-0.4,1.4],1e-4)
```

得到如图 4.12 所示曲线。从图 4.12 中可看出,在 $x = 0.5$ 附近采样点十分密集。

也可以直接用 fplot 函数绘制 $f(x) = \cos(\tan(\pi x))$ 的曲线,命令如下:

```
fplot('cos(tan(pi * x))',[-0.4,1.4],1e-4)
```

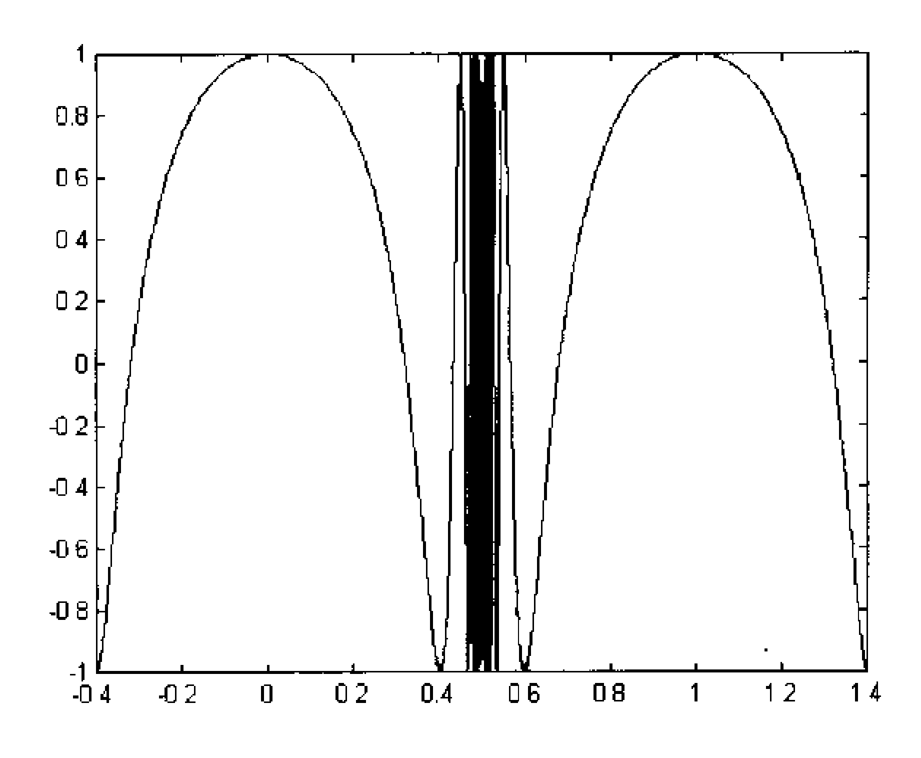


图 4.12 自适应采样绘图

5. 其他形式的图形

MATLAB 提供的绘图函数还有很多,例如,用来表示各元素占总和的百分比的饼图、复数的相量图,等等。下面以一个实例加以说明。

例 4.12 绘制图形:

(1) 某次考试优秀、良好、中等、及格、不及格的人数分别为:7,17,23,19,5,试用饼图作成绩统计分析。

(2) 绘制复数的相量图:3 + 2i、4.5 - i 和 - 1.5 + 5i。

程序如下:

```
subplot(1,2,1);
pie([7,17,23,19,5]);
title('饼图');legend('优秀','良好','中等','及格','不及格');
subplot(1,2,2);
compass([3 + 2i,4.5 - i, - 1.5 + 5i]);title('相量图');
```

程序执行结果如图 4.13 所示。

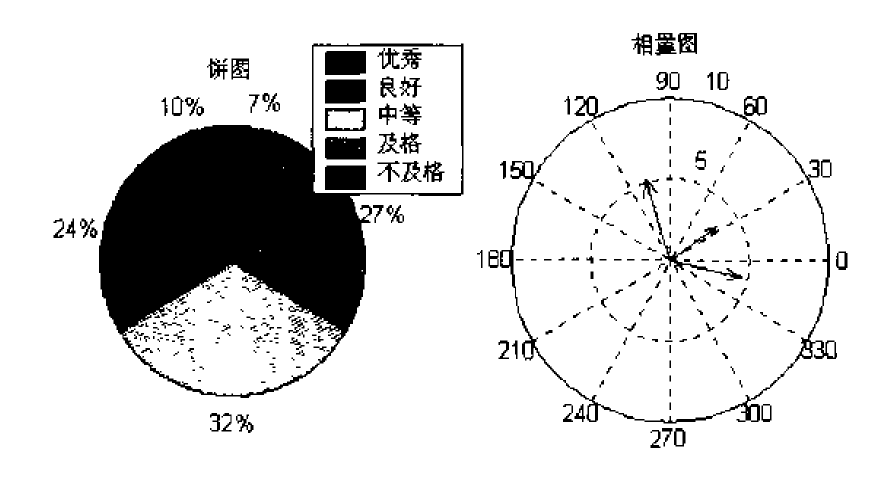


图 4.13 其他形式图形举例

4.2 三维图形

4.2.1 绘制三维曲线的最基本函数

最基本的三维图形函数为 plot3,它是将二维绘图函数 plot 的有关功能扩展到三维空间,用来绘制三维曲线。plot3 函数与 plot 函数用法十分相似,其调用格式为:

```
plot3(x1,y1,z1,选项 1,x2,y2,z2,选项 2,…,xn,yn,zn,选项 n)
```

其中每一组 x,y,z 组成一组曲线的坐标参数,选项的定义和 plot 函数相同。当 x,y,z 是同维向量时,则 x,y,z 对应元素构成一条三维曲线。当 x,y,z 是同维矩阵时,则以 x,y,z 对应列元素绘制三维曲线,曲线条数等于矩阵列数。

例 4.13 绘制空间曲线:

$$\begin{cases} x^2+y^2+z^2=64 \\ y+z=0 \end{cases}$$

曲线所对应的参数方程为:

$$\begin{cases} x=8\cos t \\ y=4\sqrt{2}\sin t, \quad 0\leqslant t\leqslant 2\pi \\ z=-4\sqrt{2}\sin t \end{cases}$$

程序如下:

```
t=0:pi/50:2*pi;
x=8*cos(t);y=4*sqrt(2)*sin(t);z=-4*sqrt(2)*sin(t);
plot3(x,y,z,'p');
title('Line in 3-D Space');text(0,0,0,'origin');
xlabel('X'),ylabel('Y'),zlabel('Z');grid;
```

结果如图 4.14 所示。

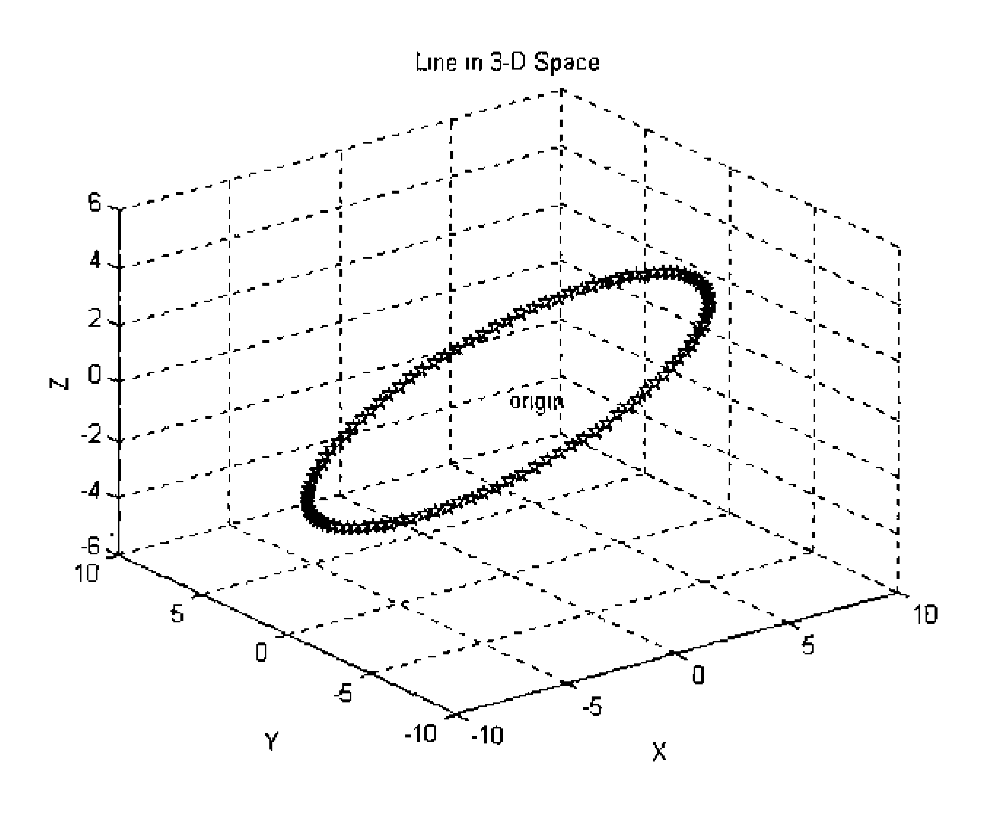


图 4.14 三维曲线

4.2.2 三维曲面

1. 平面网格坐标矩阵的生成

绘制 $z=f(x,y)$所代表的三维曲面图,先要在 $x-y$ 平面选定一矩形区域,假定矩形区域 $D=[a,b]\times[c,d]$,然后将$[a,b]$在 x 方向分成 m 份,将$[c,d]$在 y 方向分成 n 份,由各划分点分别作平行于两坐标轴的直线,将区域 D 分成 $m\times n$ 个小矩形,生成代表每一个小矩形顶点坐标的平面网格坐标矩阵,最后利用有关函数绘图。

产生平面区域内的网格坐标矩阵有两种方法:

(1) 利用矩阵运算生成,命令如下。

```
x = a:dx:b; y = (c:dy:d)';
X = ones(size(y)) * x;
Y = y * ones(size(x));
```

上述语句执行后,矩阵 **X** 的每一行都是向量 x,行数等于向量 y 的元素的个数,矩阵 **Y** 的每一列都是向量 y,列数等于向量 x 的元素的个数。于是 **X** 和 **Y** 相同位置上的元素($X(i,j)$, $Y(i,j)$)恰好是区域 D 的(i,j)网格点的坐标。若根据每一个网格点上的 x,y 坐标求函数值 z,则得到函数值矩阵 **Z**。显然,**X**,**Y**,**Z** 各列或各行所对应坐标,对应于一条空间曲线,空间曲线的集合组成空间曲面。

(2) 利用 meshgrid 函数生成,命令如下。

```
x = a:dx:b; y = c:dy:d;
[X,Y] = meshgrid(x,y);
```

语句执行后,所得到的网格坐标矩阵 **X**,**Y** 与方法(1)得到的相同。当 $x=y$ 时,meshgrid 函

数可写成 meshgrid(x)。

为了说明网格坐标矩阵的用法下面举一个例子，该例子巧妙地利用网格坐标矩阵来解不定方程。

例 4.14　已知$6 < x < 30, 15 < y < 36$，求不定方程 $2x + 5y = 126$ 的整数解。

程序如下：

```
x = 5:29; y = 14:35;
[x,y] = meshgrid(x,y);          %在[5,29]×[14,35]区域生成网格坐标
z = 2 * x + 5 * y;
k = find(z = = 126);             %找出解的位置
x(k),y(k)                        %输出对应位置的 x,y 即方程的解
```

输出为：

```
ans =
      8     13     18     23     28
ans =
      22    20     18     16     14
```

即方程共有 5 组解：(8,22)、(13,20)、(18,18)、(23,16)、(28,14)。

2. 绘制三维曲面的函数

MATLAB 提供了 mesh 函数和 surf 函数来绘制三维曲面图。mesh 函数用于绘制三维网格图。在不需要绘制特别精细的三维曲面图时，可以通过三维网格图来表示三维曲面。surf 用于绘制三维曲面图，各线条之间的补面用颜色填充。surf 函数和 mesh 函数的调用格式为：

```
mesh(x,y,z,c)
surf(x,y,z,c)
```

一般情况下，x,y,z 是维数相同的矩阵。x,y 是网格坐标矩阵，z 是网格点上的高度矩阵，c 用于指定在不同高度下的颜色范围。c 省略时，MATLAB 认为 $c = z$，亦即颜色的设定是正比于图形的高度的，这样就可以得出层次分明的三维图形。当 x,y 省略时，把 z 矩阵的列下标当作 x 轴坐标，把 z 矩阵的行下标当作 y 轴坐标，然后绘制三维曲面图。当 x,y 是向量时，必须要求 x 的长度等于 z 矩阵的列，y 的长度等于 z 矩阵的行，x,y 向量元素的组合构成网格点的 $x-y$ 坐标，z 坐标则取自 z 矩阵，然后绘制三维曲面图。

例 4.15　用三维曲面图表现函数 $z = \sin(y)\cos(x)$。

为便于分析各种三维曲面的特征，下面画出了 3 种不同形式的曲面。

程序 1：

```
x = 0:0.1:2 * pi;[x,y] = meshgrid(x);z = sin(y) . * cos(x);
mesh(x,y,z);xlabel(' x - axis '),ylabel(' y - axis '),zlabel(' z - axis ');title(' mesh ');
```

程序 2：

```
x = 0:0.1:2 * pi;[x,y] = meshgrid(x);z = sin(y) . * cos(x);
surf(x,y,z);xlabel(' x - axis '),ylabel(' y - axis '),zlabel(' z - axis ');title(' surf ');
```

程序 3：

```
x = 0:0.1:2 * pi;[x,y] = meshgrid(x);z = sin(y) . * cos(x);
plot3(x,y,z);xlabel(' x - axis '),ylabel(' y - axis '),zlabel(' z - axis ');title(' plot3 - 1 ');grid;
```

程序执行结果分别如图 4.15、4.16、4.17 所示。从图中可以发现，网格图(mesh)中线条有颜色，线条间补面无颜色。曲面图(surf)的线条是黑色，线条间补面有颜色。还可进一步观察到曲面图补面颜色和网格图线条颜色都是沿 z 轴变化的。用 plot3 绘制的三维曲面实际上由三维曲线组合而成。读者可分析一下 plot3(x',y',z')所绘制曲面的特征。

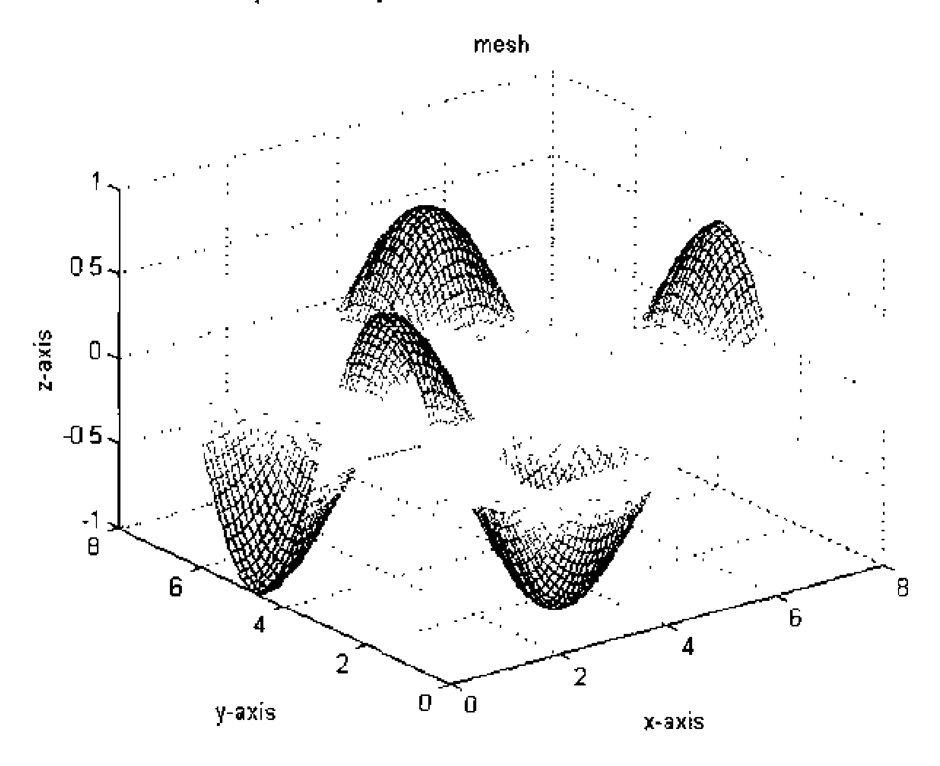


图 4.15 三维网格图

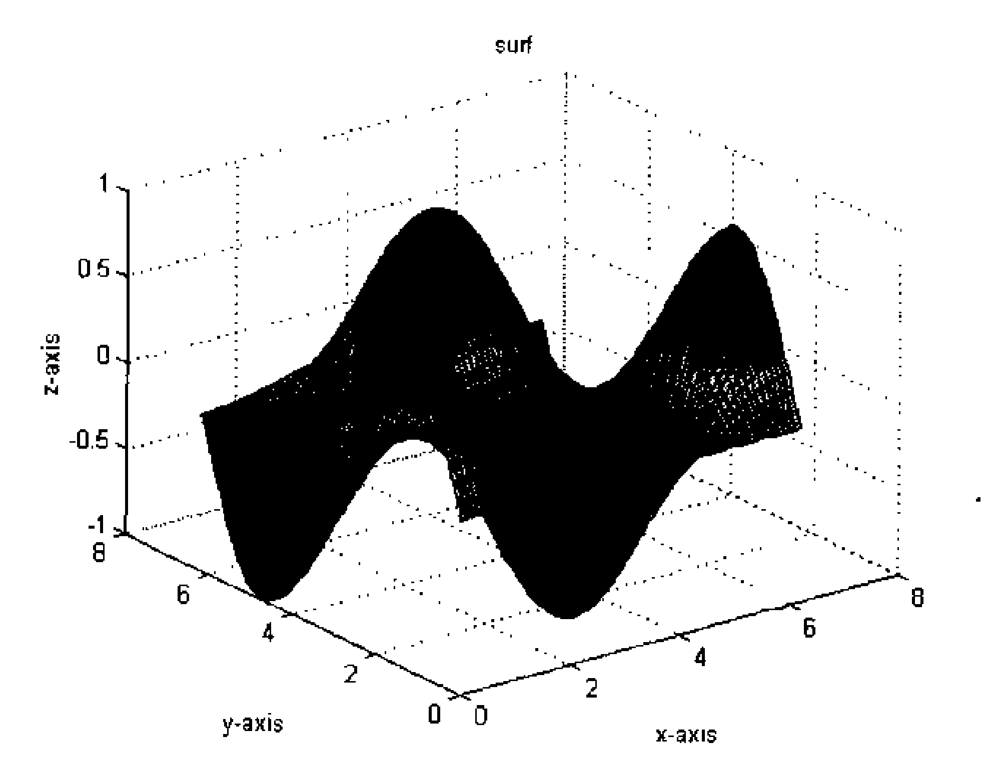


图 4.16 三维曲面图

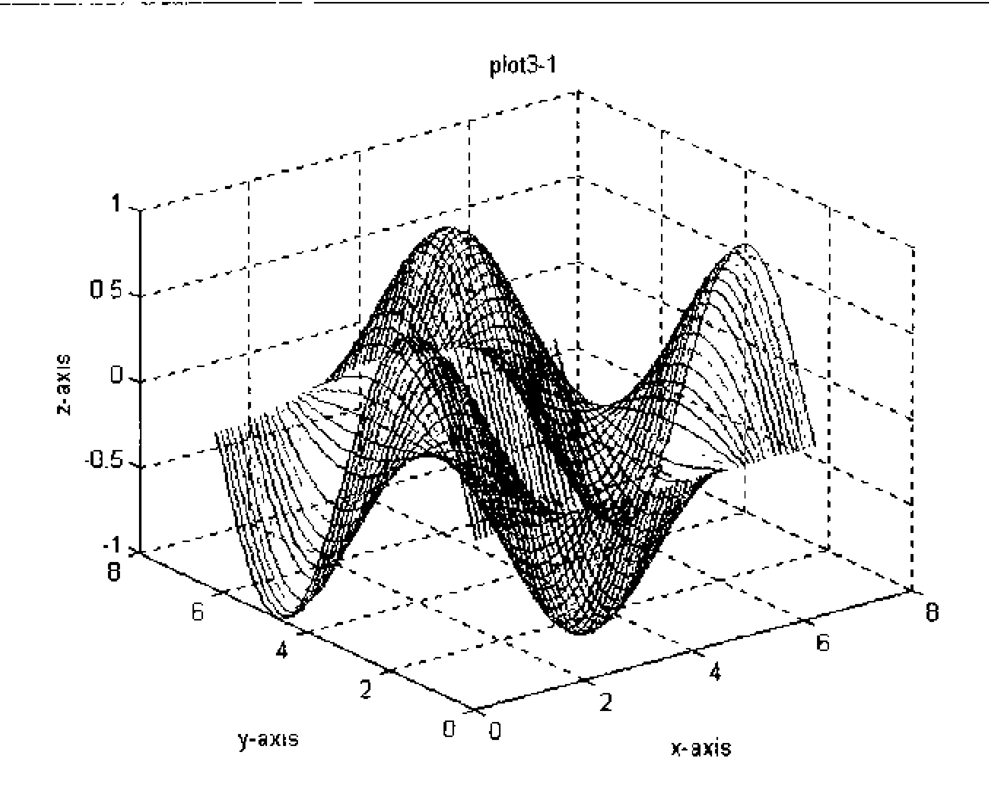


图 4.17 用 plot3 绘制的曲面图

例 4.16 分析由函数 $z=x^2-2y^2$ 构成的曲面形状及与平面 $z=a$ 的交线。

程序如下：

```
[x,y] = meshgrid( - 10:0.2:10);
z1 = (x.^2 - 2 * y.^2) + eps;                                   %第 1 个曲面
a = input(' a = ? '); z2 = a * ones(size(x));                    %第 2 个曲面
subplot(1,2,1);mesh(x,y,z1);hold on;mesh(x,y,z2);                %分别画出两个曲面
v = [ - 10,10, - 10,10, - 100,100];axis(v);grid;                 %第 1 子图的坐标设置
hold off;
r0 = abs(z1 - z2) < = 1;                                         %求两曲面 z 坐标差小于 1 的点
xx = r0. * x; yy = r0. * y; zz = r0. * z2;                       %求这些点上的 x,y,z 坐标,即交线坐标
subplot(1,2,2);
plot3(xx(r0 ~ = 0),yy(r0 ~ = 0),zz(r0 ~ = 0),' * ');             %在第 2 子图画出交线
axis(v);grid;                                                    %第 2 子图的坐标设置
```

程序执行时,若输入 a = －25,所得三维曲面图如图 4.18 左图所示,曲面的交线如图 4.18 右图所示。输入的 a 不同,曲面的交线就不同。

3. 标准三维曲面

MATLAB 提供了一些函数用于绘制标准三维曲面,还可以利用这些函数产生相应的绘图数据,常用于三维图形的演示。例如,sphere 函数和 cylinder 函数分别用于绘制三维球面和柱面。sphere 函数的调用格式为:

```
[x,y,z] = sphere(n)
```

该函数将产生 $(n+1)\times(n+1)$ 矩阵 x,y,z,采用这 3 个矩阵可以绘制出圆心位于原点、半

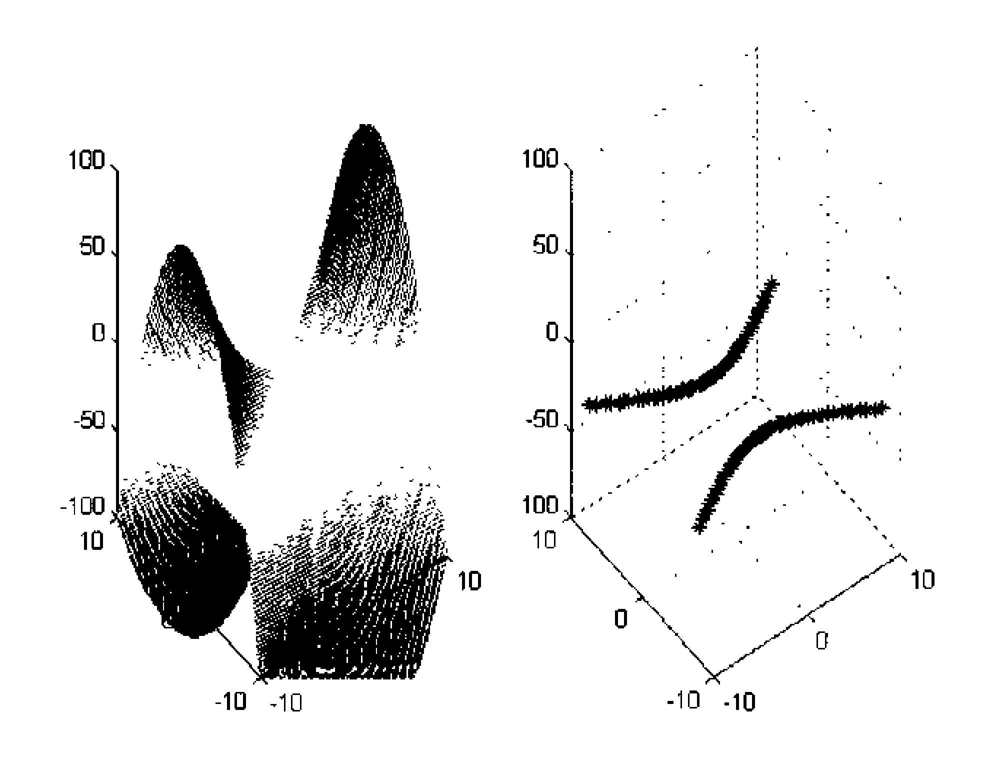


图4.18 两曲面及其交线

径为1的单位球体。若在调用该函数时不带输出参数,则直接绘制所需球面。n 决定了球面的圆滑程度,其缺省值为20。若 n 值取得较小,则将绘制出多面体表面图。

cylinder函数的调用格式为:

```
[x,y,z] = sphere(R,n)
```

其中 $\boldsymbol{R}$ 是一个向量,存放柱面各个层次上的半径。例如,cylinder(3)生成一个圆柱,cylinder([10,1])生成一个圆锥,而 t = 0:pi/100:4 * pi;R = sin(t); cylinder(R,30) 生成一个正弦型柱面。另外,生成矩阵的大小与 $\boldsymbol{R}$ 向量的长度及 n 有关。其余与sphere函数相同。

MATLAB还有一个peaks函数,称为多峰函数,常用于三维曲面的演示。该函数可以用来生成绘图数据矩阵,矩阵元素由函数

$$f(x,y)=3(1-x^2)e^{-x^2-(y+1)^2}-10\left(\frac{x}{5}-x^3-y^5\right)e^{-x^2-y^2}-\frac{1}{3}e^{-(x+1)^2-y^2}$$

在矩形区域[-3,3]×[-3,3]的等分网格点上的函数值确定。例如

```
z = peaks (30);
```

将生成一个30×30矩阵 z,即分别沿 x 和 y 方向将区间[-3,3]等分成29份,并计算这些网格点上的函数值。缺省的等分数是48,即 p = peaks 将生成一个49×49矩阵p。也可以根据网格坐标矩阵 x,y 重新计算函数值矩阵,例如

```
[x,y] = meshgrid( -5:0.1:5);
z = peaks(x,y);
```

生成的数值矩阵可以作为mesh、surf等函数的参数而绘制出多峰函数曲面图。另外,若在调用peaks函数时不带输出参数,则直接绘制出多峰函数曲面图。

4.2.3 其他三维图形

在介绍二维图形时,曾提到条形图、饼图和填充图等特殊图形,它们还可以以三维形式出现,使用的函数分别是 bar3、pie3 和 fill3。此外,还有三维曲面的等高线图。等高线图分二维和三维两种形式,分别使用函数 contour 和 contour3 绘制。下面举例说明三维等高线图的绘制。

例 4.17 绘制多峰函数的等高线图。

程序如下:

```
[x,y,z] = peaks;
contour3(x,y,z,12,'k');      %其中 12 代表高度的等级数
xlabel('x-axis'),ylabel('y-axis'),zlabel('z-axis');
title('contour3 of peaks');
```

程序执行结果如图 4.19 所示。

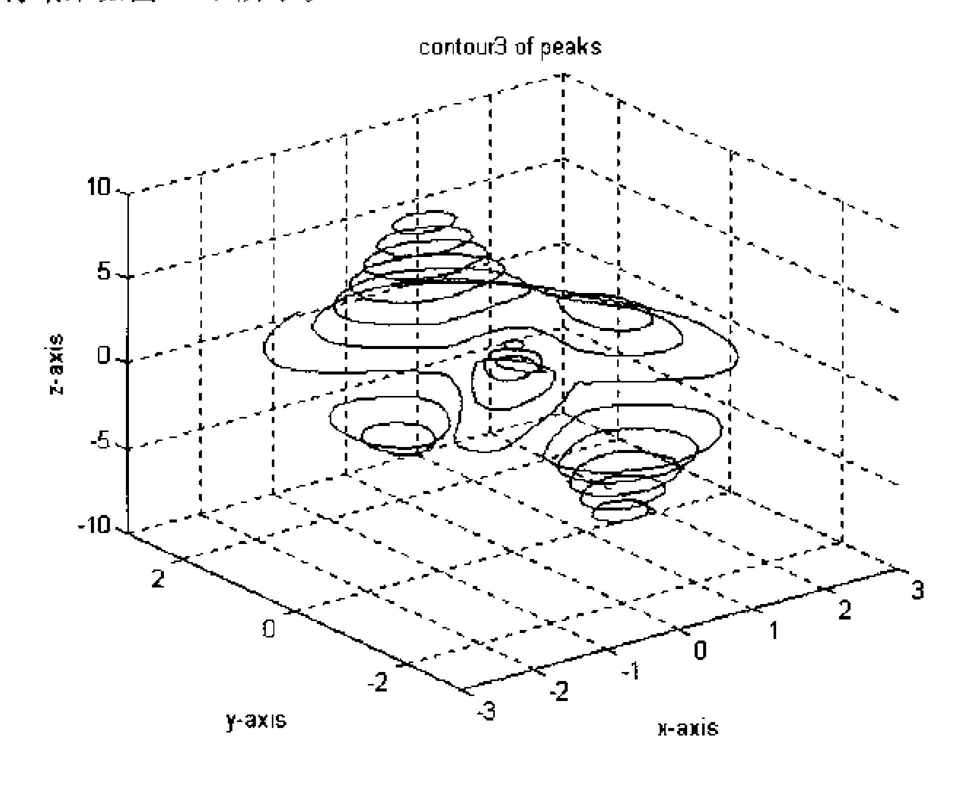


图 4.19 三维等高线图

4.3 三维图形的精细处理

4.3.1 图形的裁剪处理

MATLAB 定义的 NaN 常数可以用于表示那些不可使用的数据,利用这种特性,可以将图形中需要裁剪部分对应的函数值设置成 NaN,这样在绘制图形时,函数值为 NaN 的部分将不显示出来,从而达到对图形进行裁剪的目的。

例 4.18　裁掉例4.15三维曲面图中 $z>0.25$ 的部分。

程序如下：

```
x = 0:0.1:2 * pi;[x,y] = meshgrid(x);z = sin(y) .* cos(x);
[I,J] = find(z > 0.25);
for ii = 1:length(I)
    z(I(ii),J(ii)) = NaN;
end
surf(x,y,z);
```

程序执行结果如图 4.20 所示。

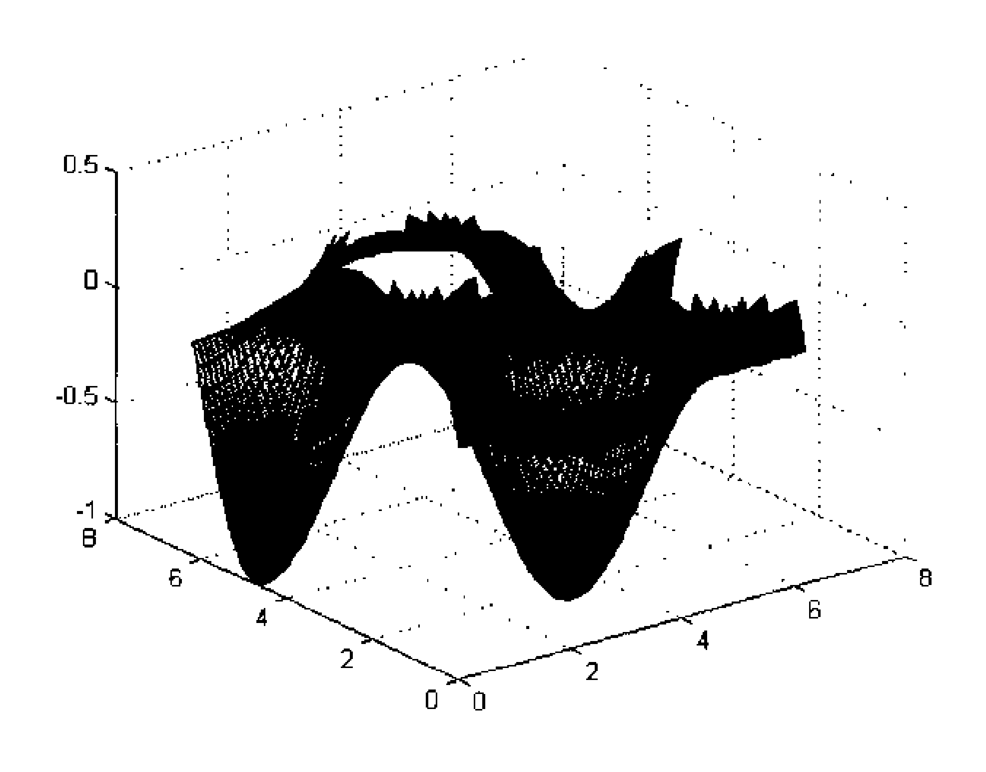


图 4.20　图形裁剪

4.3.2　视点处理

在日常生活中，从不同的视点观察物体，所看到的物体形状是不一样的。同样，从同不视点绘制的三维图形其形状也是不一样的。视点位置可由方位角和仰角表示。方位角又称旋转角，它是视点与原点连线在 $x-y$ 平面上的投影与 y 轴负方向形成的角度，正值表示逆时针，负值表示顺时针。仰角又称视角，它是视点与原点连线与 $x-y$ 平面的夹角，正值表示视点在 $x-y$ 平面上方，负值表示视点在 $x-y$ 平面下方。图 4.21 示意了坐标系中视点的定义，图中箭头方向表示正的方向。

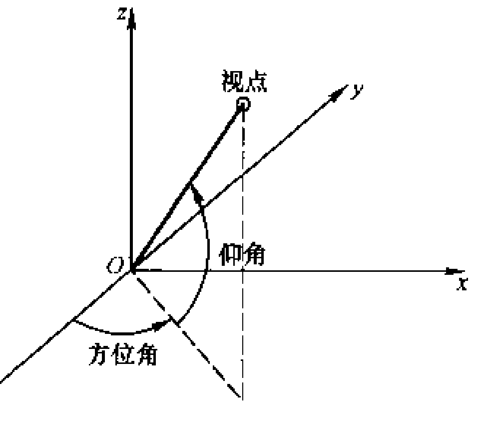


图 4.21　视点的定义

从视点的定义不难看出，仰角为 90°或 -90°表示视点在 z 轴。仰角为 0°表示视点在 $x-y$ 平面上，此时从 x 轴正方向

看方向角为 90°,从 y 轴负方向看方向角为 0°。据此,通过视点的设置可以绘制出三维曲面在不同平面上的投影。MATLAB 提供了设置视点的函数 view。其调用格式为:

```
view(az,el)
```

其中 az 为方位角,el 为仰角,它们均以度为单位。系统缺省的视点定义为方位角 -37.5°,仰角 30°。

例 4.19 从不同视点绘制多峰函数曲面。

程序如下:

```
subplot(2,2,1);mesh(peaks);
view(-37.5,30);                    %指定子图 1 的视点
title('azimuth=-37.5,elevation=30')
subplot(2,2,2);mesh(peaks);
view(0,90);                        %指定子图 2 的视点
title('azimuth=0,elevation=90')
subplot(2,2,3);mesh(peaks);
view(90,0);                        %指定子图 3 的视点
title('azimuth=90,elevation=0')
subplot(2,2,4);mesh(peaks);
view(-7,-10);                      %指定子图 4 的视点
title('azimuth=-7,elevation=-10')
```

程序执行结果如图 4.22 所示,该图充分反映了视点对图形的影响。

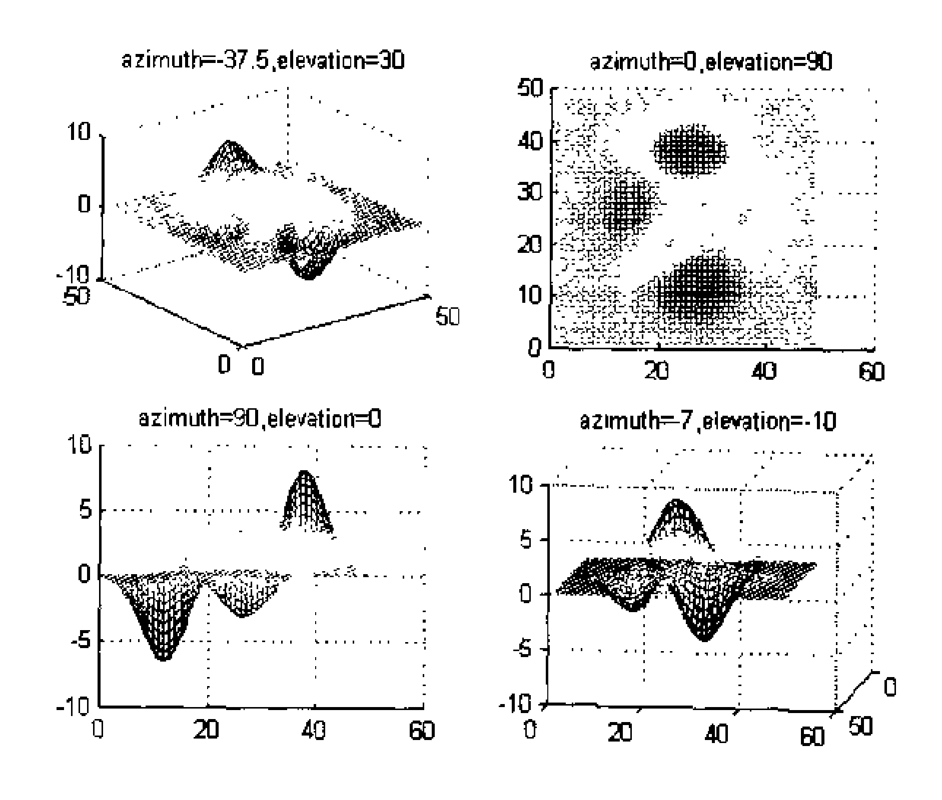


图 4.22 不同视点图形

4.3.3 色彩处理

1. 颜色的向量表示

MATLAB 除用字符表示颜色外(见表4.1),还可以用含有3个元素的向量表示颜色。向量元素在[0,1]范围取值,3个元素分别表示红、绿、蓝3种颜色的相对亮度,称为 RGB 三元组。表4.2中列出了几种常见颜色的 RGB 值。

表4.2 几种常见颜色的 RGB 值

RGB 值	颜色	字符	RGB 值	颜色	字符
[0 0 1]	蓝色	b	[1 1 1]	白色	w
[0 1 0]	绿色	g	[0.5 0.5 0.5]	灰色	
[1 0 0]	红色	r	[0.67 0 1]	紫色	
[0 1 1]	青色	c	[1 0.5 0]	橙色	
[1 0 1]	品红色	m	[1 0.62 0.40]	铜色	
[1 1 0]	黄色	y	[0.49 1 0.83]	宝石蓝	
[0 0 0]	黑色	k			

2. 色图

色图(Color map)是 MATLAB 系统引入的概念。在 MATLAB 中,每个图形窗口只能有一个色图。色图是 $m \times 3$ 的数值矩阵,它的每一行是 RGB 三元组。色图矩阵可以人为地生成,也可以调用 MATLAB 提供的函数来定义色图矩阵。表4.3列出了定义色图矩阵的函数,色图矩阵的维数由函数调用格式决定。例如

表4.3 定义色图矩阵函数

函数名	含义	函数名	含义
autumn	红、黄浓淡色	jet	蓝头红尾饱和值色
bone	蓝色调浓淡色	lines	采用 plot 绘线色
colorcube	三浓淡多彩交错色	pink	淡粉红色图
cool	青、品红浓淡色	prism	光谱交错色
copper	纯铜色调线性浓淡色	spring	青、黄浓淡色
flag	红—白—蓝—黑交错色	summer	绿、黄浓淡色
gray	灰色调线性浓淡色	winter	蓝、绿浓淡色
hot	黑、红、黄、白浓淡色	white	全白色
hsv	两端为红的饱和值色		

```
M = hot;
```

生成 64×3 色图矩阵 $\boldsymbol{M}$,表示的是从黑色、红色、黄色到白色的由浓到淡的颜色。又如

```
P = gray(100);
```

生成 100×3 色图矩阵 $\boldsymbol{P}$,表示的是灰色由浓到淡的颜色。

除 plot 及其派生函数外,mesh、surf 等函数均使用色图着色。图形窗口色图的设置和改变,使

用函数：

```
colormap(m)
```

其中 m 代表色图矩阵。

3. 三维表面图形的着色

三维表面图实际上就是在网格图的每一个网格片上涂上颜色。surf 函数用缺省的着色方式对网格片着色。除此之外，还可以用 shading 命令来改变着色方式。

(1) shading faceted 命令　将每个网格片用其高度对应的颜色进行着色，但网格线仍保留着，其颜色是黑色。这是系统的缺省着色方式。

(2) shading flat 命令　将每个网格片用同一个颜色进行着色，且网格线也用相应的颜色，从而使得图形表面显得更加光滑。

(3) shading interp 命令　在网格片内采用颜色插值处理，得出的表面图显得最光滑。

例 4.20　3种图形着色方式的效果展示。

程序如下：

```
z = peaks(20);colormap(copper);
subplot(1,3,1);surf(z);
subplot(1,3,2); surf(z);shading flat;
subplot(1,3,3);surf(z);shading interp;
```

程序执行结果如图 4.23 所示。

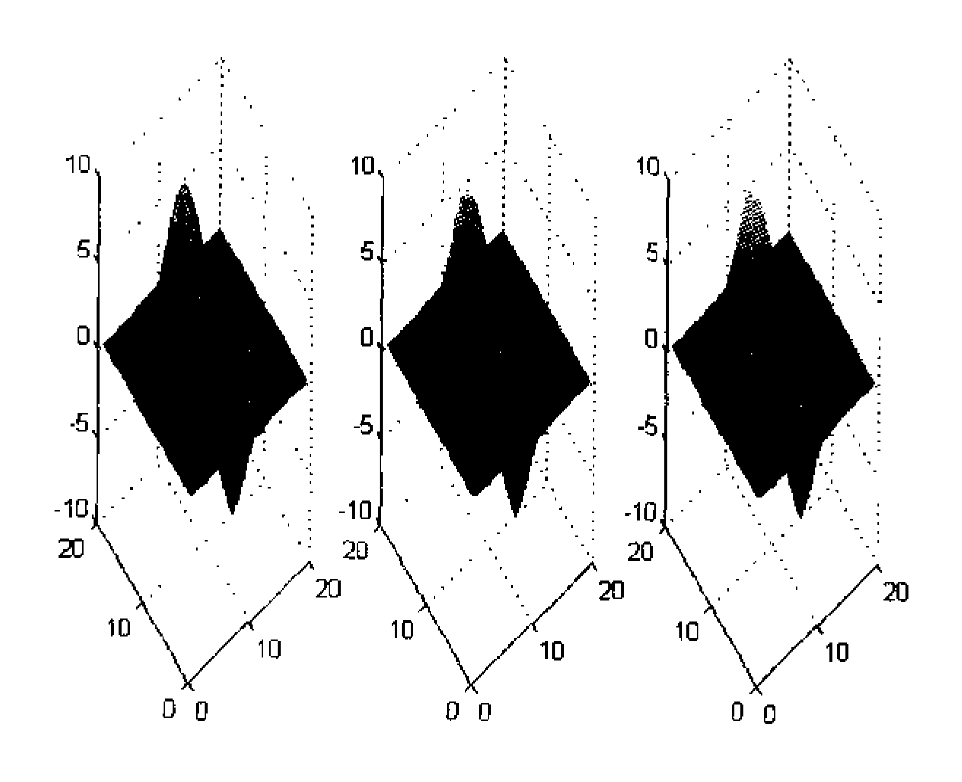


图 4.23　图形着色

4.3.4 光照处理

使用光照处理将把图形表现得更加真实。MATLAB 提供了灯光设置的函数,其调用格式为:

```
light('Color',选项 1,'Style',选项 2,'Position',选项 3)
```

其中选项 1 表示光的颜色,取 RGB 三元组或相应的颜色字符。选项 2 有'infinite'和'local'两个取值,分别表示无穷远光和近光。选项 3 取三维坐标点组成的向量形式[x,y,z]。对远光,它表示穿过该点射向原点;对于近光,它表示光源所在位置。假如函数不包含任何参数,则采用缺省设置:白光、无穷远、穿过(1,0,1)点射向坐标原点。

例 4.21　光照处理后的多峰函数曲面。

程序如下:

```
z = peaks(20);
subplot(1,2,1);surf(z);
light('Posi',[0,20,10]);shading interp;hold on;
plot3(0,20,10,'p');text(0,20,10,'light');
subplot(1,2,2);surf(z);
light('Posi',[20,0,10]);shading interp;hold on;
plot3(20,0,10,'p');text(20,0,10,'light');
```

程序执行结果如图 4.24 所示。

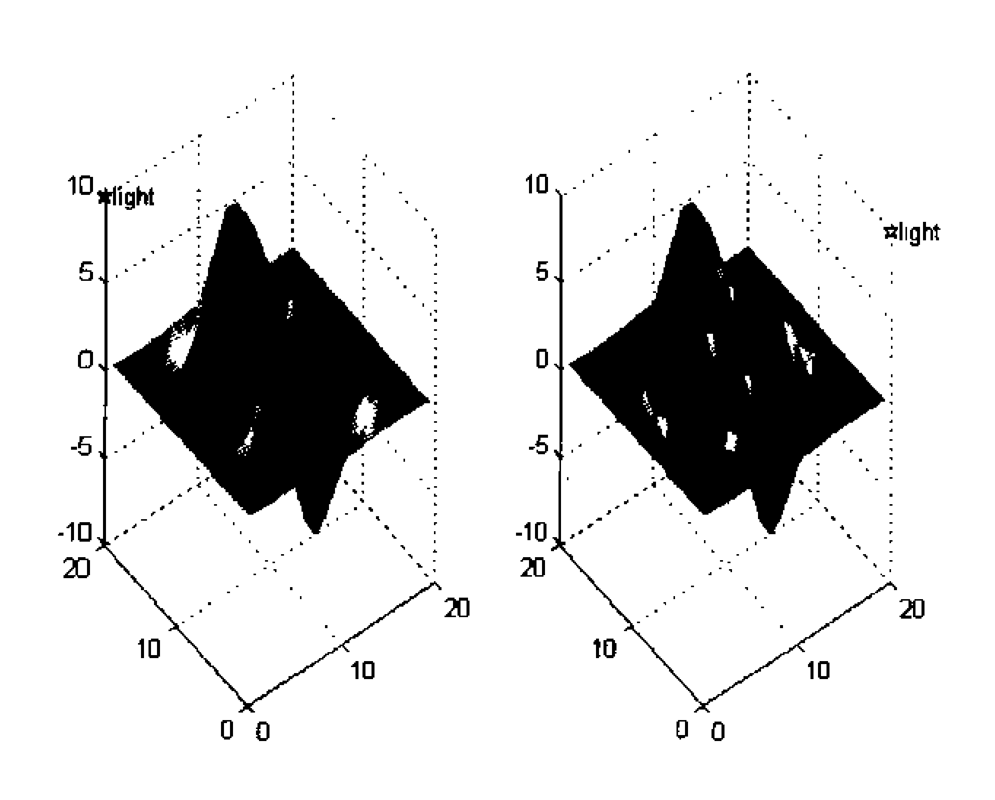


图 4.24　光照处理后的图形

4.4　图像与动画

4.4.1　图像

MATLAB 基本系统提供了几个用于简单图像处理的函数，利用这些函数可进行图像的读写和显示。此外，MATLAB 还有一个功能更强的图像处理工具箱，可以对图像进行更专业的处理。

1. imread 和 imwrite 函数

imread 和 imwrite 函数分别用于将图像文件读入 MATLAB 工作空间，以及将图像数据和色图数据一起写入一定格式的图像文件。MATLAB 支持多种图像文件格式，如 .bmp、.jpg、.jpeg、.tif 等。

2. image 和 imagesc 函数

这两个函数用于图像显示。为了保证图像的显示效果，一般还应使用 colormap 函数设置图像色图。

例 4.22　在 E 盘根目录下有一图像文件 building.jpg，在图形窗口显示该图像。

程序如下：

```
[x,cmap] = imread('e:\building.jpg');   %读取图像的数据阵和色图阵
image(x);colormap(cmap);
axis image off   %保持宽高比并取消坐标轴
```

执行结果如图 4.25 所示。



图 4.25　图像显示

4.4.2 动画

如果将 MATLAB 产生的多幅图形保存起来,并利用系统提供的函数进行播放,就可产生动画效果。系统所提供的动画功能函数有 getframe、moviein 和 movie。

1. getframe 函数

getframe 函数可截取每一幅画面信息而形成一个很大的列向量。该向量可保存到一个变量中。显然,保存 *n* 幅图就需一个大矩阵。

2. moviein 函数

moviein(n)函数用来建立一个足够大的 *n* 列矩阵。该矩阵用来保存 *n* 幅画面的数据,以备播放。之所以要事先建立一个大矩阵,是为了提高程序运行速度。

3. movie 函数

movie(m,n)函数以每秒 *n* 幅图的速度播放由矩阵 *m* 的列向量所组成的画面。

例 4.23　播放一个直径不断变化的球体。

程序如下:

```
[x,y,z] = sphere(50);
m = moviein(30);                %建立一个 30 列大矩阵
for i = 1:30
    surf(i * x,i * y,i * z)     %绘制球面
    m(:,i) = getframe;          %将球面保存到 m 矩阵
end
movie(m,10);                    %以每秒 10 幅的速度播放球面
```

4.5 低层绘图操作

迄今为止,本书所介绍的都是 MATLAB 高层绘图函数。绘图本来是一项很繁琐的工作,需要确定很多参数,但 MATLAB 高层绘图函数代替用户做了这些工作,给定了参数的缺省值,这样就使用户免去了一些操作细节,用起来很方便。但一旦遇到缺省值不能满足实际需要时,就需要用户的干预。低层绘图操作就能解决这个问题。

4.5.1 图形对象及其句柄

1. 图形对象

MATLAB 把构成图形的各个基本要素称为图形对象。这些对象包括计算机屏幕、图形窗口(Figure)、坐标轴(Axes)、用户菜单(Uimenu)、用户控件(Uicontrol)、曲线(Line)、曲面(Surface)、文字(Text)、图像(Image)、光源(Light)、区域块(Patch)和方框(Rectangle)。系统将每一个对象按树型结构组织起来(图 4.26)。每个图形对象都可以被独立地操作。

在 MATLAB 中,每一个具体的图形都是由若干个不同的图形对象组成。每个具体图形不一定包含全部对象,但必须带有计算机屏幕和图形窗口对象。

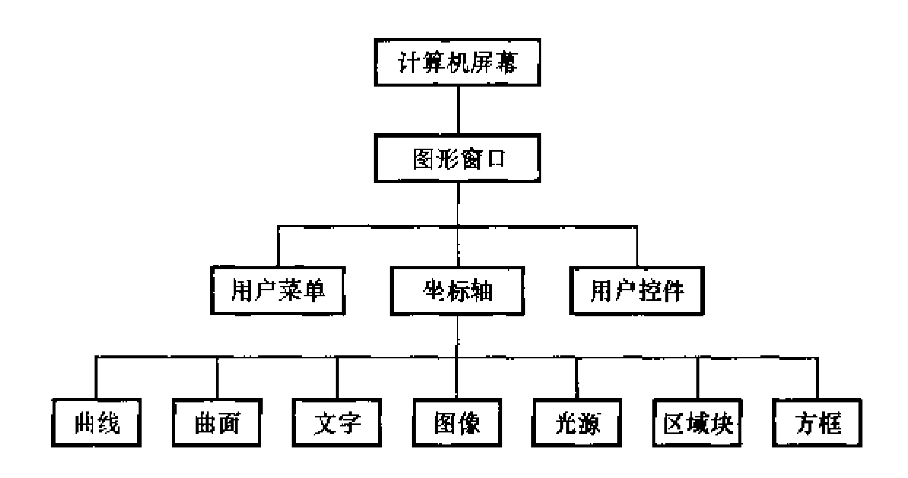


图 4.26　图形对象的树型结构

计算机屏幕是产生其他对象的基础，称之为根对象。它包含一个或多个图形窗口对象。一个图形窗口对象有 3 种不同类型的子对象：坐标轴、用户菜单和用户控件。其中后两类对象用于构建图形用户界面，第 7 章将做专门介绍。坐标轴有 7 种不同类型的子对象：曲线、曲面、文字、图像、光源、区域块、方框。对坐标轴及其 7 种子对象的操作即构成低层绘图操作，本节介绍其主要内容。

2. 图形对象句柄

MATLAB 在创建每一个图形对象时，都为该对象分配惟一的一个值，称其为图形对象句柄(Handle)。句柄是图形对象的惟一标识符，不同对象的句柄不可能重复和混淆。

计算机屏幕作为根对象由系统自动建立，其句柄值为 0，而图形窗口对象的句柄值为一正整数，并显示在该窗口的标题栏，其他图形对象的句柄为浮点数。MATLAB 提供了 3 个用于获取已有图形对象句柄的函数：

gcf　获取当前图形窗口的句柄(get current figure)。

gca　获取当前坐标轴的句柄(get current axis)。

gco　获取最近被单击的图形对象的句柄(get current object)。

为了以后对图形对象进行操作，可以在建立该对象时保存其句柄。例如，hf = figure 命令用 figure 函数建立一个图形窗口对象，并将其句柄赋给变量 hf 保存，以后就可以通过该句柄对图形窗口进行操作。

4.5.2　图形对象属性

1. 属性名与属性值

每种图形对象都具有各种各样的属性，MATLAB 正是通过对属性的操作来控制和改变图形对象的。为方便属性的操作，MATLAB 给每种对象的每一个属性规定了一个名字，称为属性名，而属性名的取值称为属性值。例如，LineStyle 是曲线对象的一个属性名，它的值决定着线型，取值可以是 -、:、-.、-- 或 none。在属性名的写法中，不区分字母的大小写，而且在不引起歧义的前提下，属性名不必写全。例如，lines 就代表 LineStyle。此外，属性名要用单撇号括起来。

2. 属性的操作

当创建一个对象时，必须给对象的各种属性赋予必要的属性值，否则，系统自动使用缺省属

性值。用户可以通过 set 函数重新设置对象属性,同时也可通过 get 函数获取这些属性值。

set 函数的调用格式为:

```
set(句柄,属性名1,属性值1,属性名2,属性值2,…)
```

其中句柄用于指明要操作的对象。如果在调用 set 函数时省略全部属性名和属性值,则将显示出句柄所有的允许属性。

曾经提及,绘制二维曲线时,通过选择不同的选项可以设置曲线的颜色、线型和数据点的标记符号,下面用图形句柄操作来实现。假定要绘制正弦曲线,命令如下:

```
x = 0:pi/10:2 * pi;
h = plot(x,sin(x));
set(h,'Color','r','LineStyle',':','Marker','p');
```

先用缺省属性绘制正弦曲线并保存曲线句柄,然后通过改变曲线的属性来设置曲线的颜色、线型和数据点的标记符号。事实上,还有很多其他属性,通过改变这些属性,可对曲线作更进一步的控制。

get 函数的调用格式为:

```
V = get(句柄,属性名)
```

其中 V 是返回的属性值。如果在调用 get 函数时省略属性名,则将返回句柄所有的属性值。

用 get 函数来获得上述曲线的属性值。例如

```
col = get(h,'Color');
```

将得到曲线的颜色属性值[1 0 0],即红色。

用 get 函数可获取屏幕的分辨率:

```
V = get(0,'ScreenSize')
```

此时,get 函数将返回一个 1×4 的向量 V,其中前两个分量分别是屏幕的左下角横纵坐标(1,1),而后两个分量分别是屏幕的宽度和高度。假如屏幕分辨率设置为 800×600,则 V 的值为[1 1 800 600]。这有助于依据现行屏幕的分辨率来设置图形窗口的大小。

3. 对象的公共属性

图形对象具有各种各样的属性,有些属性是所有对象共同具备的,有些则是各对象所特有的。这里先介绍对象常用的公共属性。

(1) Children 属性。该属性的取值是该对象所有子对象的句柄组成的一个向量。

(2) Parent 属性。该属性的取值是该对象的父对象的句柄。显然,图形窗口对象的 Parent 属性总是 0。

(3) Tag 属性。该属性的取值是一个字符串,它相当于给该对象定义了一个标识符。定义了 Tag 属性后,在任何程序中都可以通过 findobj 函数获取该标识符所对应图形对象的句柄。例如,hf = findobj(0,'Tag','Flag1')将在屏幕对象及其子对象中寻找 Tag 属性为 Flag1 的对象,并返回句柄。

(4) Type 属性。表示该对象的类型。显然,该属性的值是不可改变的。

(5) UserData 属性。该属性的取值是一个矩阵,缺省值为空矩阵。在程序设计中,可以将一个图形对象有关的比较重要的数据存储在这个属性中,借此可以达到传递数据的目的。具体做法是,先用 set 函数给某一句柄添加一些附加数据(一个矩阵),如果想使用这样的矩阵,则用 get

函数调用出来。

(6) Visible 属性。该属性的取值是 on(缺省值)或 off。决定着图形窗口是否在屏幕上显示出来。当它的值为 off 时,可以用来隐藏该图形窗口的动态变化过程,如窗口大小的变化、颜色的变化等。注意,对象存在与否与对象是否可见是两回事,对象可以存在,同时又是不可见的。

(7) ButtonDownFcn 属性。该属性的取值是一个字符串,一般是某个 M 文件名或一小段 MATLAB 语句。图形对象决定了一个作用区域,当单击该区域时,MATLAB 自动执行该程序段。

(8) CreateFcn 属性。该属性的取值是一个字符串,一般是某个 M 文件名或一小段 MATLAB 语句。当创建该对象时,MATLAB 自动执行该程序段。

(9) DeleteFcn 属性。该属性的取值是一个字符串,一般是某个 M 文件名或一小段 MATLAB 语句。当取消该对象时,MATLAB 自动执行该程序段。

例 4.24　在同一坐标下画红、绿两根不同曲线,希望获得绿色曲线的句柄,并对其进行设置。

程序如下:

```
x = 0:pi/50:2 * pi;y = sin(x);z = cos(x);
plot(x,y,'r',x,z,'g');
Hl = get(gca,'Children');                    %获取两曲线句柄向量 Hl
for k = 1:size(Hl)
    if get(Hl(k),'Color') == [0 1 0]         %[0 1 0]代表绿色
        Hlg = Hl(k);                         %获取绿色曲线句柄
    end
end
pause                                        %为便于观察设置效果而增加本语句,可不加
set(Hlg,'LineStyle',':');                    %对绿色曲线进行设置
```

4.5.3　图形对象的创建

除根对象外,所有图形对象都可以由与之同名的低层函数创建。所创建的对象置于适当的父对象之中,当父对象不存在时,MATLAB 会自动创建它。例如,用 line 函数画一根曲线,假如在画线之前,坐标轴、图形窗口不存在,MATLAB 会自动创建它们。假如在画线之前,坐标轴、图形窗口已经存在,那么将在当前坐标轴上画线,且不影响该坐标轴上已有的其他对象。这一点与高层绘图函数完全不同,需特别注意。

创建对象的各低层函数调用格式类似,关键要了解对象的属性及其取值。前面介绍了各对象的公共属性,下面介绍常用图形对象的创建方法及特殊属性。

1. 图形窗口对象

图形窗口是 MATLAB 中很重要的一类图形对象。MATLAB 的一切图形图像的输出都是在图形窗口中完成的。掌握好图形窗口的控制方法,对于充分发挥 MATLAB 的图形功能和设计高质量的用户界面是十分重要的。

建立图形窗口对象使用 figure 函数。调用该函数的命令形式为:

句柄变量 = figure(属性名 1,属性值 1,属性名 2,属性值 2,…)

MATLAB 通过对属性的操作来改变图形窗口的形式。也可以使用 figure 函数按 MATLAB 缺省的属性值建立图形窗口:

figure 或 句柄变量 = figure

MATLAB 通过 figure 函数建立窗口之后，还可以调用 figure 函数来显示该窗口，并将之设定为当前窗口，命令如下：

figure(窗口句柄)

其实，即使这里引用的窗口句柄不存在，也可以使用这一命令，它的作用是对这一窗口句柄生成一个新的窗口，并将之定义为当前窗口。

函数“close(窗口句柄)”将关闭相应的图形窗口。close all 命令可以关闭所有的图形窗口。clf 命令则是清除当前图形窗口的内容，并不关闭窗口。

MATLAB 为每个图形窗口提供了很多属性。这些属性及其取值控制着图形窗口对象。除公共属性外，其他常用属性如下：

(1) MenuBar 属性。该属性的取值可以是 figure(缺省值)或 none。用来控制图形窗口是否应该具有菜单条。如果它的属性值为 none，则表示该图形窗口没有菜单条。这时用户可以采用 uimenu 函数来加入自己的菜单条，如果属性值为 figure，则该窗口将保持图形窗口默认的菜单条，这时也可以采用 uimenu 函数在原默认的图形窗口菜单后面添加新的菜单项。

(2) Name 属性。该属性的取值可以是任何字符串，它的缺省值为空。这个字符串作为图形窗口的标题。一般情况下，其标题形式为：Figure No.1：字符串。

(3) NumberTitle 属性。该属性的取值是 on(缺省值)或 off。决定着在图形窗口的标题中是否以“Figure No.n:”为标题前缀，这里 n 是图形窗口的序号，即句柄值。

(4) Resize 属性。该属性的取值是 on(缺省值)或 off。决定着在图形窗口建立后可否用鼠标改变该窗口的大小。

(5) Position 属性。该属性的取值是一个由 4 个元素构成的向量，其形式为[n1,n2,n3,n4]。这个向量定义了图形窗口对象在屏幕上的位置和大小，其中 $n1$ 和 $n2$ 分别为窗口左下角的横纵坐标值，$n3$ 和 $n4$ 分别为窗口的宽度和高度。它们的单位由 Units 属性决定。

(6) Units 属性。该属性的取值可以是下列字符串中的任何一种：pixel(像素，为缺省值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)和 points(磅)。

Units 属性定义图形窗口使用的长度单位，由此决定图形窗口的大小与位置。除了 normalized 以外，其他单位都是绝对度量单位。相对单位 normalized 将屏幕左下角对应为(0,0)，而右上角对应为(1.0,1.0)。该属性将影响一切定义大小的属性项，如前面的 Position 属性。如果在程序中改变过 Units 属性值，在完成相应的计算后，最好将 Units 属性值设置为缺省值，以防止影响其他函数计算。

(7) Color 属性。该属性的取值是一个颜色值，既可以用字符表示，也可以用 RGB 三元组表示。用于控制图形窗口的背景颜色缺省值为黑色。

(8) Pointer 属性。该属性的取值是 arrow(缺省值)、crosshair、watch、topl、topr、botl、botr、circle、cross、fleur、custom 等，用于设定鼠标标记的显示形式。

(9) 对键盘及鼠标响应属性。MATLAB 允许对类似于键盘和鼠标键按下这样的动作进行响应，这类属性有 KeyPressFcn(键盘键按下响应)、WindowButtonDownFcn(鼠标键按下响应)、WindowButtonMotionFcn(鼠标移动响应)及 WindowButtonUpFcn(鼠标键释放响应)等，这些属性所对应的属性值可以为用 MATLAB 编写的函数名或命令名，表示一旦键盘键或鼠标键按下之后，将自动调用给出的函数或命令。

例 4.25　建立一个图形窗口。该图形窗口没有菜单条,标题名称为"图形窗口示例",起始于屏幕左下角、宽度和高度分别为 300 像素和 150 像素,背景颜色为绿色,且当用户从键盘按下任意一个键时,将显示"Hello,Keyboard Key Pressed."字样。

命令如下:

```
hf = figure('Color',[0,1,0],'Position',[1,1,300,150],...
    'Name','图形窗口示例','NumberTitle','off','MenuBar','none',...
    'KeyPressFcn','disp(''Hello,Keyboard Key Pressed.'')');
```

例 4.26　分别在4个不同的图形窗口绘制出正弦、余弦、正切、余切曲线。要求先建立一个图形窗口并绘图,然后每关闭一个再建立下一个,直到建立第 4 个窗口并绘图。

程序如下:

```
x = linspace(0,2*pi,60);
y = sin(x);z = cos(x);t = tan(x);ct = 1./(t+eps);
%命令组待用
C4 = ['figure(''Name'',''cotangent(x)'',''NumberTitle'',',...
    '''off'');plot(x,ct);axis([0,2*pi,-40,40]);'];
C3 = ['figure(''Name'',''tangent(x)'',''DeleteFcn'',C4,',...
    '''NumberTitle'',''off'');plot(x,t);axis([0,2*pi,-40,40]);'];
C2 = ['figure(''Name'',''cos(x)'',''DeleteFcn'',C3,',...
    '''NumberTitle'',''off'');plot(x,z);axis([0,2*pi,-1,1]);'];
%先创建 1 个图形窗口并绘制曲线
figure('Name','sin(x)','DeleteFcn',C2,'NumberTitle','off');
plot(x,y);axis([0,2*pi,-1,1]);
```

2. 坐标轴对象

坐标轴是 MATLAB 中另一类很重要的图形对象。坐标轴对象是图形窗口的子对象,每个图形窗口中可以定义多个坐标轴对象,但只有一个坐标轴是当前坐标轴,在没有指明坐标轴时,所有的图形图像都是在当前坐标轴中输出。必须弄清一个概念,所谓在某个图形窗口中输出图形图像,实质上是指在该图形窗口的当前坐标轴中输出图形图像。

建立坐标轴对象使用 axes 函数,调用它的命令形式为:

```
句柄变量 = axes(属性名 1,属性值 1,属性名 2,属性值 2,…)
```

调用 axes 函数用指定的属性在当前图形窗口创建坐标轴,并将其句柄赋给左边的句柄变量。也可以使用 axes 函数按 MATLAB 缺省的属性值在当前图形窗口创建坐标轴:

```
axes　或　句柄变量 = axes
```

用 axes 函数建立坐标轴之后,还可以调用 axes 函数将之设定为当前坐标轴,且坐标轴所在的图形窗口自动成为当前图形窗口:

```
axes(坐标轴句柄)
```

注意,这里引用的坐标轴句柄必须存在,这与 figure 函数的对应调用形式不同。

MATLAB 为每个坐标轴对象提供了很多属性。除公共属性外,其他常用属性如下:

(1) Box 属性。该属性的取值是 on 或 off(缺省值)。它决定坐标轴是否带有边框。

(2) GridLineStyle 属性。该属性的取值可以是:(缺省值)、- 、- .、- - 或 none。该属性定义网格线的类型。

(3) Position 属性。该属性的取值是一个由 4 个元素构成的向量，其形式为[n1,n2,n3,n4]。这个向量在图形窗口中决定一个矩形区域，坐标轴就位于其中。该矩形的左下角相对于图形窗口左下角的坐标为($n1$,$n2$)、矩形的宽和高分别为 $n3$ 和 $n4$。它们的单位由 Units 属性决定。

(4) Units 属性。该属性的取值是 normalized(相对单位，为缺省值)、inches(英寸)、centimeters(厘米)和 points(磅)。Units 属性定义 Position 属性的度量单位。

(5) Title 属性。该属性的取值是坐标轴标题文字对象的句柄，可以通过该属性对坐标轴标题文字对象进行操作。例如，要改变标题的颜色，可执行命令：

```
set(get(gca,'Title'),'Color','r')
```

(6) XLabel、YLabel、ZLabel 属性。3 种属性的取值分别是 x,y,z 轴说明文字的句柄。

(7) XLim、YLim、ZLim 属性。3 种属性的取值都是 2 个元素的数值向量。三属性分别定义各坐标轴的上下限，缺省值为[0,1]。以前介绍的 axis 函数实际上是对这些属性的直接赋值。

(8) XScale、YScale、ZScale 属性。3 种属性的取值都是 linear(缺省值)或 log，这些属性定义各坐标轴的刻度类型。

(9) View 属性。该属性的取值是 2 个元素的数值向量，定义视点方向。

例 4.27　利用坐标轴对象实现图形窗口的任意分割。

利用 axes 函数可以在不影响图形窗口上其他坐标轴的前提下建立一个新的坐标轴，从而实现图形窗口的任意分割。程序如下：

```
clf;  %清图形窗口
x = linspace(0,2 * pi,20);y = sin(x);
axes('Posi',[0.2,0.2,0.2,0.7]);plot(y,x);title('sin(x) - 1');
axes('Posi',[0.4,0.5,0.2,0.1]);stairs(x,y);title('sin(x) - 2');
axes('Posi',[0.55,0.6,0.25,0.3]);stem(x,y);title('sin(x) - 3');
axes('Posi',[0.55,0.2,0.25,0.3]);stem(x,y);title('sin(x) - 4');
```

程序执行结果如图 4.27 所示。

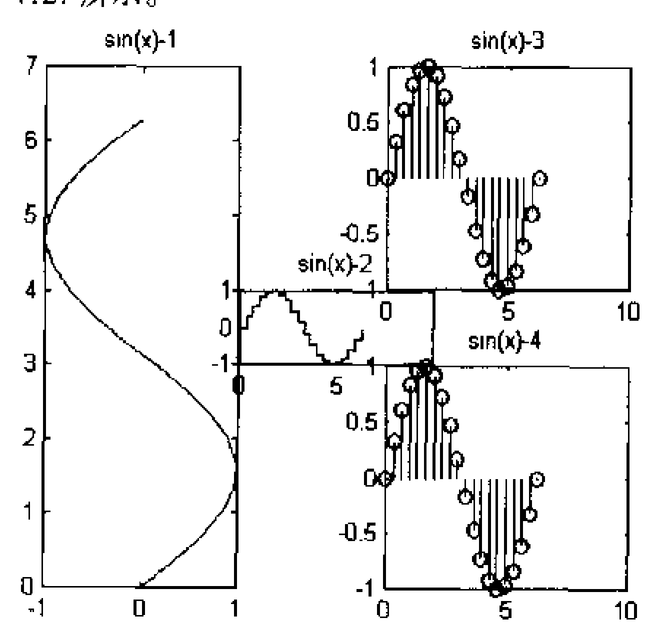


图 4.27　图形窗口的任意分割

3. 曲线对象

曲线对象是坐标轴的子对象，它既可以定义在二维坐标系中，也可以定义在三维坐标系中。建立曲线对象使用 line 函数，调用它的命令形式为：

```
句柄变量 = line(x,y,z,属性名 1,属性值 1,属性名 2,属性值 2,…)
```

其中对 x,y,z 的解释与高层曲线函数 plot 和 plot3 等一样，其余的解释与前面介绍过的 figure 和 axes 函数类似。

每个曲线对象也具有很多属性。除公共属性外，其他常用属性如下：

(1) Color 属性。定义曲线的颜色。

(2) LineStyle 属性。定义线型。

(3) LineWidth 属性。定义线宽。缺省值为 0.5 磅。

(4) Marker 属性。定义数据点标记符号。缺省值为 none。

(5) MarkerSize 属性。定义数据点标记符号的大小。缺省值为 6 磅。

(6) XData、YData、ZData 属性。3 种属性的取值都是数值向量或矩阵。分别代表曲线对象的 3 个坐标轴数据。

4. 文字对象

文字对象主要用于给图形添加文字标注。MATLAB 从 5.0 版本开始，对文字对象作了重大改进，在文字对象中除使用一般的文字以外，还允许使用 LATEX(LATEX 是一种十分流行的数学排版软件)文本，这样就可以在图形上添加希腊字母、数学符号及公式等内容。

使用 text 函数可以根据指定位置和属性值添加文字说明，并保存句柄。调用该函数的命令形式为：

```
句柄变量 = text(x,y,z,'说明文字',属性名 1,属性值 1,属性名 2,属性值 2,…)
```

其中说明文字中除使用标准的 ASCII 字符外，还可使用 LATEX 格式的控制字符。例如

```
h = text(0.5,0.5,'x^2 + sin({\omega}t + {\beta})');
```

将得到标注效果：$x^2 + \sin(\omega t + \beta)$。

除公共属性外，文字对象的其他常用属性如下：

(1) Color 属性。定义文字对象的显示颜色。

(2) String 属性。该属性的取值是字符串或字符串矩阵，它记录着文字标注的内容。

(3) Interpreter 属性。该属性的取值是 latex(缺省值)或 none，该属性控制对文字标注内容的解释方式，即 LATEX 方式或 ASCII 方式。

(4) FontSize 属性。定义文字对象的大小。缺省值为 10 磅。

(5) Rotation 属性。该属性的取值是数值量，缺省值为 0。它定义文字对象的旋转角度，取正值时表示逆时针方向旋转，取负值时表示顺时针方向旋转。

例 4.28　利用曲线对象完成例4.5。

程序如下：

```
x = (0:pi/100:2 * pi)';
y1 = 2 * exp( - 0.5 * x) * [1, - 1];
y2 = 2 * exp( - 0.5 * x) .* sin(2 * pi * x);
x1 = (0:12)/2;
y3 = 2 * exp( - 0.5 * x1) .* sin(2 * pi * x1);
```

```
line(x,y1,'LineStyle',':','color','g');
line(x,y2,'LineStyle','--','color','b');
line(x1,y3,'LineStyle','none','Marker','p','color','r');
title('曲线及其包络线');
xlabel('independent variable X');
ylabel('independent variable Y');
text(2.8,0.55,'2e^{-0.5x}','FontSize',12);
text(0.45,0.55,'y=2e^{-0.5x}sin(2{\pi}x)','FontSize',12);
text(1.4,0.1,'离散数据点');
legend('包络线','包络线','曲线 y','离散数据点')
```

程序执行结果如图 4.28 所示。

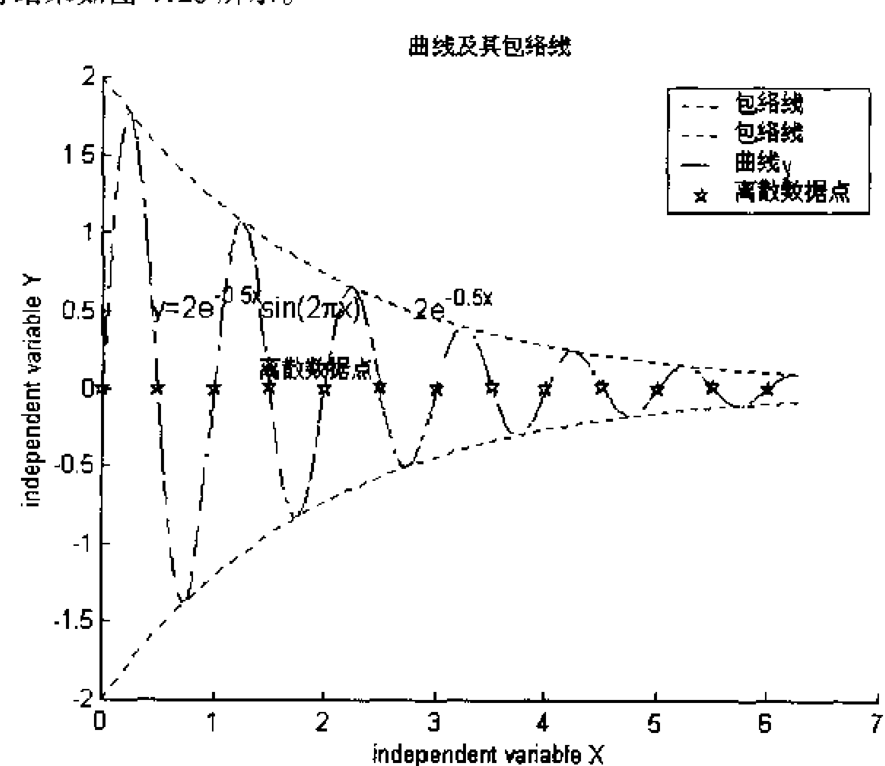


图 4.28 利用曲线对象绘制的曲线

5. 曲面对象

曲面对象也是坐标轴的子对象，它定义在三维坐标系中，而坐标系可以在任何视点下。建立曲面对象使用 surface 函数，调用它的命令形式为：

句柄变量 = surface(x,y,z,属性名 1,属性值 1,属性名 2,属性值 2,…)

其中对 x,y,z 的解释与高层曲面函数 mesh 和 surf 等一样，其余的解释与前面介绍过的 figure 和 axes 等函数类似。

每个曲面对象也具有很多属性。除公共属性外，其他常用属性如下：

(1) EdgeColor 属性。该属性的取值是代表某颜色的字符或 RGB 值，还可以是 flat、interp 或 none。缺省为黑色。定义曲面网格线的颜色或着色方式。

(2) FaceColor 属性。该属性的取值是代表某颜色的字符或 RGB 值，还可以是 flat(缺省值)、

interp 或 none。定义曲面网格片的颜色或着色方式。

(3) LineStyle 属性。定义曲面网格线的线型。

(4) LineWidth 属性。定义曲面网格线的线宽。缺省值为 0.5 磅。

(5) Marker 属性。定义曲面数据点标记符号。缺省值为 none。

(6) MarkerSize 属性。定义曲面数据点标记符号的大小。缺省值为 6 磅。

(7) XData、YData、ZData 属性。3 种属性的取值都是数值向量或矩阵。分别代表曲面对象的 3 个坐标轴数据。

例 4.29 利用曲面对象绘制三维曲面 $z=\sin(y)\cos(x)$。

程序如下：

```
x = 0:0.1:2 * pi;[x,y] = meshgrid(x);z = sin(y) . * cos(x);
axes(' view ',[ - 37.5,30]);
hs = surface(x,y,z,' FaceColor ',' w ',' EdgeColor ',' flat ');
grid on;xlabel(' x - axis '),ylabel(' y - axis '),zlabel(' z - axis ');title(' mesh - surf ');
pause;
set(hs,' FaceColor ',' flat ');
```

开始网格片的颜色设为白色,实际上得到的是网格图,这与高层函数 mesh 所画曲面相同(见图 4.15)。暂停后,重新设置网格片的颜色,得到着色表面图(图 4.29 所示)。

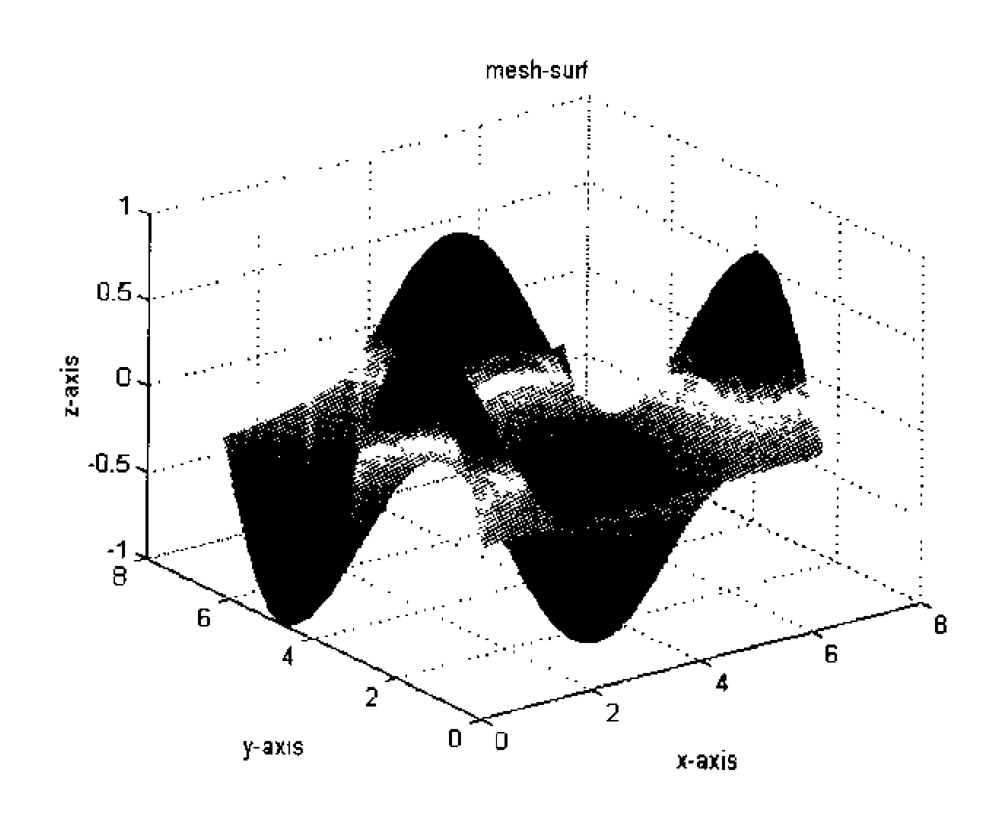


图 4.29 利用曲面对象绘制的曲面

习　题　四

1．绘制下列曲线。

(1) $y=\frac{100}{1+x^2}$　　(2) $y=\frac{1}{2\pi}e^{-\frac{x^2}{2}}$

(3) $x^2+y^2=1$　　(4) $\begin{cases}x=t^2\\y=5t^3\end{cases}$

2．绘制下列极坐标图。

(1) $\rho=5\cos\theta+4$　　(2) $\rho=\frac{12}{\sqrt{\theta}}$

(3) $\rho=\frac{5}{\cos\theta}-7$　　(4) $\rho=\frac{\pi}{3}\theta^2$

3．绘制下列三维图形。

(1) $\begin{cases}x=\cos t\\y=\sin t\\z=t\end{cases}$　　(2) $z=5$

(3) $z=\frac{x^2}{2}+\frac{y^2}{3}$　　(4) $z=\frac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}}$

4．分别用 plot 和 fplot 函数绘制下列分段函数的曲线。

$$f(x)=\begin{cases}x^2+\sqrt[4]{1+x}+5, & x>0\\0, & x=0\\x^3+\sqrt{1-x}-5, & x<0\end{cases}$$

5．绘制例 2.5 经过处理后的正弦波。

6．某工厂 2000 年度各季度产值(单位:万元)分别为:450.6、395.9、410.2、450.9,试绘制折线图和饼图,并说明图形的实际意义。

7．根据$\frac{x^2}{a^2}+\frac{y^2}{25-a^2}=1$绘制平面曲线,并分析参数 a 对其形状的影响。

8．低层绘图操作的基本思路是什么？它同高层绘图操作相比有何特点？

第 5 章　MATLAB 数值计算

数值计算在科学研究与工程应用中有非常广泛的应用。许多数值计算问题,用其他程序设计语言编程求解非常麻烦,并且需要具备专门的数学知识及一定的程序设计技能,而用 MATLAB 编程,往往只要很少几个语句即可完成求解任务,而且它编程效率高、对程序员的数学背景知识要求低。MATLAB 这种强大的数值计算能力,使其在科学计算方面成为了用户的首选工具。

本章先介绍线性代数中的数值计算问题,包括特殊矩阵、矩阵分析、线性方程组求解等内容,然后介绍数据处理和多项式计算,包括数据的分析与统计、数据插值、曲线拟合及多项式计算等内容,最后介绍数值微积分、常微分方程数值解以及非线性代数方程求解等内容。

5.1 特殊矩阵

本节在矩阵基本运算的基础上,讨论特殊矩阵及其实现问题,包括矩阵的三角提取、对角提取以及特殊矩阵生成等内容。

5.1.1 对角阵与三角阵

1. 矩阵的对角元素

矩阵的第一行第一列和第 p 行 p 列组成的一条斜线称为主对角线,有时把主对角线就简称为对角线,把与主对角线垂直的另一条对角线称为次对角线。只有对角线上有非 0 元素的矩阵称为对角矩阵简称对角阵,对角线上的元素相等的对角矩阵称为标量矩阵或数量矩阵,对角线上的元素都为 1 的对角矩阵称为单位矩阵。矩阵的对角线有许多性质:如转置运算时对角线元素不变,相似变换时对角线上的元素的和(称为矩阵的迹)不变等。在研究矩阵时,很多时候需要将矩阵的对角线上的元素提取出来形成一个列向量,而有时又需要用一个向量构造一个对角阵。

(1) 提取矩阵的对角线元素

设 $\boldsymbol{A}$ 为 $m\times n$ 矩阵,diag(A)函数提取矩阵 $\boldsymbol{A}$ 主对角线元素,产生一个具有 min(m,n)个元素的列向量。例如,在 MATLAB 命令窗口输入命令,得到的结果如下:

```
A = [10,20,30;40,50,60];
D = diag(A)
D =
    10
    50
```

diag(A)函数还有更进一步的形式 diag(A,k),其功能是提取第 k 条对角线的元素。与主对角线平行,往上为第 1 条、第 2 条、…、第 n 条对角线,往下为第 -1 条、第 -2 条、…、第 $-n$ 条对角线。主对角线为第 0 条对角线。例如,对于上面建立的 $\boldsymbol{A}$ 矩阵,提取其主对角线两侧对角线

的元素,命令及运行结果如下:

```
D1 = diag(A,1)
D =
    20
    60
D2 = diag(A, - 1)
D =
    40
```

(2) 构造对角矩阵

设 V 为具有 m 个元素的向量,diag(V)将产生一个 $m \times m$ 的对角矩阵,其主对角线元素即为向量 V 的元素。例如,在 MATLAB 命令窗口输人命令,及得到的结果如下:

```
diag([1,2, - 1,4])
ans =
      1     0     0     0
      0     2     0     0
      0     0   - 1     0
      0     0     0     4
```

diag(V)函数也有更进一步的形式 diag(V,k),其功能是产生一个 $n \times n(n = m + |k|)$对角阵,其第 k 条对角线的元素即为向量 V 的元素。例如

```
diag(1:3, - 1)
ans =
      0     0     0     0
      1     0     0     0
      0     2     0     0
      0     0     3     0
```

例 5.1　先建立 5×5 矩阵 A,然后将 A 的第 1 行元素乘以 1,第 2 行乘以 2,…,第 5 行乘以 5。

用一个对角矩阵左乘一个矩阵时,相当于用对角阵的第 1 个元素乘以该矩阵的第 1 行,用对角阵的第 2 个元素乘以该矩阵的第 2 行,…,依次类推,因此只需要按要求构造一个对角矩阵 D,并用 D 左乘 A 即可。命令如下及运行结果:

```
A = [17,0,1,0,15;23,5,7,14,16;4,0,13,0,22;10,12,19,21,3;11,18,25,2,19];
D = diag([1,2,3,4,5]);
D * A                          %用 D 左乘 A,对 A 的每行乘以一个指定常数
ans =
        17     0     1     0    15
        46    10    14    28    32
        12     0    39     0    66
        40    48    76    84    12
        55    90   125    10    95
```

如果要对 A 的每列元素乘以同一个数,可以用一个对角阵右乘矩阵 A。

2. 矩阵的三角阵

三角矩阵简称三角阵,三角阵又进一步分为上三角阵和下三角阵,所谓上三角阵,即矩阵的

对角线以下的元素全为 0 的一种矩阵，而下三角阵则是对角线以上的元素全为 0 的一种矩阵。

(1) 下三角矩阵

与矩阵 **A** 对应的下三角阵 **B** 是与 **A** 同型(具有相同的行数和列数)的一个矩阵，并且 **B** 的对角线以下(含对角线)的元素和 **A** 对应相等，而对角线以上的元素等于 0。求矩阵 **A** 的下三角阵的 MATLAB 函数是 tril(A)。例如，提取矩阵 **A** 的下三角元素，形成新的矩阵 **B**，命令及运行结果如下：

```
A = [37,10,-8;2,19,8;0,34,-65];
B = tril(A)
B =
      37     0     0
       2    19     0
       0    34   -65
```

tril(A)函数也有更进一步的一种形式 tril(A,k)，其功能是求矩阵 **A** 的第 k 条对角线以下的元素。例如，提取矩阵 **A** 的第 2 条对角线以下的元素，形成新的矩阵 **B**，命令及运行结果如下：

```
A = [17,0,1,0,15;23,5,7,14,16;4,0,13,0,22;10,12,19,21,3;11,18,25,2,19];
B = tril(A,2)
B =
    17   0   1   0   0
    23   5   7  14   0
     4   0  13   0  22
    10  12  19  21   3
    11  18  25   2   9
```

(2) 上三角矩阵

在 MATLAB 中，提取矩阵 *A* 的上三角矩阵的函数是 triu(A)和 triu(A,k)，其用法与提取下三角矩阵的函数 tril(A)和 tril(A,k)完全相同，请读者自己总结该函数的基本功能和用法。

5.1.2 特殊矩阵的生成

MATLAB 中提供了一些函数，利用这些函数可以很方便地生成一些特殊矩阵，下面介绍几个常用函数的功能和用法。

1. 魔方矩阵

魔方矩阵有一个有趣的性质，其每行、每列及两条对角线上的元素和都相等。对于 n 阶魔方阵，其元素由 $1,2,3,\cdots,n^2$ 共 n^2 个整数组成。MATLAB 提供了求魔方矩阵的函数 magic(n)，其功能是生成一个 n 阶魔方阵。

例 5.2 将101～125这 25 个数填入一个 5 行 5 列的表格中，使其每行每列及对角线的和均为 565。

首先构造一个 5 阶魔方矩阵，使其每行、每列及对角线的和均为 65，然后，对其每个元素都加 100 后这些和都变为 565。命令及运行结果如下：

```
B = 100 + magic(5)
B =
```

```
117  124  101  108  115
123  105  107  114  116
104  106  113  120  122
110  112  119  121  103
111  118  125  102  109
```

2. 范得蒙德矩阵

下面是一个范得蒙德(Vandermonde)矩阵的实例：

$$A=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \\ 27 & 9 & 3 & 1 \\ 125 & 25 & 5 & 1 \end{bmatrix}$$

其最后一列全为 1,倒数第二列为一个指定的向量,其他各列是其后列与倒数第二列的乘积。可以用一个指定向量生成一个范得蒙德矩阵。在 MATLAB 中,函数 vander(V)生成以向量 V 为基础向量的范得蒙德矩阵。例如,A = vander([1;2;3;5])即可得到上述范得蒙德矩阵。

3. 希尔伯特矩阵

希尔伯特(Hilbert)矩阵的每个元素 $h_{ij}=\dfrac{1}{i+j-1}$。在 MATLAB 中,生成希尔伯特矩阵的函数是 hilb(n)。希尔伯特矩阵是一个条件数(详见 5.2.6 节)很差的矩阵,使用一般方法求逆会因为原始数据的微小扰动而产生不可靠的计算结果。MATLAB 中,有一个专门求希尔伯特矩阵的逆的函数 invhilb(n),其功能是求 n 阶的希尔伯特矩阵的逆矩阵。

4. 特普利茨矩阵

特普利茨(Toeplitz)矩阵除第 1 行第 1 列外,其他每个元素都与左上角的元素相同。生成特普利茨矩阵的函数是 toeplitz(x,y),它生成一个以 x 为第 1 列,y 为第 1 行的特普利茨矩阵。这里 x, y 均为向量,二者不必等长。

5. 友矩阵

设多项式 $P(x)$为：

$$P(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$$

称矩阵 A

$$A=\begin{bmatrix} -\dfrac{a_{n-1}}{a_n} & -\dfrac{a_{n-2}}{a_n} & -\dfrac{a_{n-3}}{a_n} & \cdots & -\dfrac{a_1}{a_n} & -\dfrac{a_0}{a_n} \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

为多项式 $P(x)$的友矩阵。在 5.2.7 中将指出：$P(x)$称为 A 的特征多项式,方程 $P(x)=0$ 的根称为 A 的特征值。

生成友矩阵的函数是：compan(P),生成多项式 P 的友矩阵。P 是一个多项式的系数向量,高次幂系数排在前,低次幂排在后。

6. 帕斯卡矩阵

我们知道,二次项$(x+y)^n$展开后的系数随n的增大组成一个三角形表,称为杨辉三角形。由杨辉三角形表组成的矩阵称为帕斯卡(Pascal)矩阵。函数 pascal(n)生成一个n阶的帕斯卡矩阵。

例 5.3 求$(x+y)^5$的展开式。

MATLAB 的命令及运行结果如下:

```
pascal(6)
ans =
     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```

其次对角线上的元素 1,5,10,10,5,1 即为展开式的系数。

5.2 矩 阵 分 析

矩阵是 MATLAB 的基本处理对象,第 2 章介绍了怎样建立矩阵以及矩阵的基本运算,读者已经感受到了 MATLAB 在处理矩阵问题时带给用户的方便性与灵活性,这一节继续讨论有关矩阵的计算与性质。

5.2.1 矩阵结构变换

1. 矩阵的转置

转置是矩阵的一种最常见的变形。所谓转置,即把源矩阵的第一行变成目标矩阵第一列,第二行变成第二列,…,显然,一个m行n列的矩阵经过转置运算后,变成一个n行m列的矩阵。设$\boldsymbol{A}$为$m\times n$矩阵,则其转置矩阵$\boldsymbol{B}$的元素定义如下:

$$b_{ji}=a_{ij}\qquad i=1,2,\cdots,m,j=1,2,\cdots,n$$

转置运算符是单斜撇号(′)。例如,求矩阵$\boldsymbol{A}$的转置矩阵$\boldsymbol{B}$的命令为$\boldsymbol{B}=\boldsymbol{A}'$。

2. 矩阵的旋转

在 MATLAB 中,可以很方便地以 90°为单位对矩阵$\boldsymbol{A}$按逆时针方向进行旋转。矩阵的旋转利用函数 rot90(A,k),功能是将矩阵$\boldsymbol{A}$旋转 90°的k倍,当k为 1 时可省略。例如,将$\boldsymbol{A}$按逆时针方向旋转 90°,命令及运行结果如下:

```
A = [37,10,-8;2,19,8;0,34,-65];
B = rot90(A)
B =
    -8     8   -65
    10    19    34
    37     2     0
```

如果输入的函数是 rot90(A,4),结果应该是什么?请读者思考并上机验证自己的结论。

3. 矩阵的左右翻转

对矩阵实施左右翻转是将原矩阵的第 1 列和最后 1 列调换,第 2 列和倒数第 2 列调换,…。MATLAB 对矩阵 **A** 实施左右翻转的函数是 fliplr(A)。针对上面建立的矩阵 **A**,将其左右翻转,命令及运行结果如下:

```
B = fliplr(A)
B =
   -8   10   37
    8   19    2
  -65   34    0
```

4. 矩阵的上下翻转

与矩阵的左右翻转类似,矩阵的上下翻转是将原矩阵的第 1 行与最后 1 行调换,第 2 行与倒数第 2 行调换,…MATLAB 对矩阵 **A** 实施上下翻转的函数是 flipud(A)。

5.2.2 矩阵的逆与伪逆

1. 矩阵的逆

矩阵的逆是线性代数中的一个基本概念。对于一个方阵 **A**,如果存在一个与其同阶的方阵 **B**,使得:

$$\boldsymbol{AB} = \boldsymbol{BA} = \boldsymbol{I}\ (\boldsymbol{I}\ \text{为单位矩阵})$$

则称 **B** 为 **A** 的逆矩阵,当然,**A** 也是 **B** 的逆矩阵。

求一个矩阵的逆是一件非常烦琐的工作,容易出错,但在 MATLAB 中,求一个矩阵的逆非常容易。求方阵 **A** 的逆可调用函数 inv(A)。

例 5.4　用求逆矩阵的方法解线性方程组。

$$\begin{cases} x + 2y + 3z = 5 \\ x + 4y + 9z = -2 \\ x + 8y + 27z = 6 \end{cases}$$

设

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 5 \\ -2 \\ 6 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

则原线性方程组可简写为

$$\boldsymbol{Ax} = \boldsymbol{b}$$

其解为

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

有了上面的分析,就可以很方便地应用 MATLAB 求解了。命令及运行结果如下:

```
A = [1,2,3;1,4,9;1,8,27]; b = [5,-2,6]';
x = inv(A) * b
x =
   23.0000
```

```
 -14.5000
  3.6667
```

用求系数矩阵逆的方法求解线性方程组虽然简单，但 MATLAB 不推荐用户这样做。一般情况下，用左除比求矩阵的逆的方法更有效，即 x = A \ b。

2. 矩阵的伪逆

如果矩阵 $\boldsymbol{A}$ 不是一个方阵，或者 $\boldsymbol{A}$ 是一个非满秩的方阵时，矩阵 $\boldsymbol{A}$ 没有逆矩阵，但可以找到一个与 $\boldsymbol{A}$ 的转置矩阵 $\boldsymbol{A}'$ 同型的矩阵 $\boldsymbol{B}$，使得

$$\boldsymbol{ABA} = \boldsymbol{A}$$

$$\boldsymbol{BAB} = \boldsymbol{B}$$

此时称矩阵 $\boldsymbol{B}$ 为矩阵 $\boldsymbol{A}$ 的伪逆（也称为广义逆矩阵），记作 $\boldsymbol{A}^{+}$。MATLAB 中，求一个矩阵伪逆的函数是 pinv(A)。

例 5.5　求 $\boldsymbol{A}$ 的伪逆，并将结果送 $\boldsymbol{B}$。

$$\boldsymbol{A} = \begin{bmatrix} 3 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & 1 & 3 & 1 \end{bmatrix}$$

命令及运行结果如下：

```
A=[3,1,1,1;1,3,1,1;1,1,3,1];
B=pinv(A)
B =
    0.3929   -0.1071   -0.1071
   -0.1071    0.3929   -0.1071
   -0.1071   -0.1071    0.3929
    0.0357    0.0357    0.0357
```

例 5.6　设矩阵

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

求 $\boldsymbol{A}$ 的伪逆。

MATLAB 中，命令及运行结果如下：

```
A=[0,0,0;0,1,0;0,0,1];
pinv(A)
ans =
     0  0  0
     0  1  0
     0  0  1
```

本例中，$\boldsymbol{A}$ 是一个奇异矩阵，无一般意义上的逆矩阵，故本例中求 $\boldsymbol{A}$ 的伪逆矩阵。此外，例中 $\boldsymbol{A}$ 的伪逆矩阵和 $\boldsymbol{A}$ 相等，读者不要由此得出一个错误结论：矩阵的伪逆矩阵就是该矩阵本身。一般说来，矩阵的伪逆矩阵和自身是不同的，读者不妨用其他矩阵再上机试试。

5.2.3 方阵的行列式

行列式是和矩阵并列的另一个数学概念。把一个方阵看作一个行列式，并对其按行列式的规则求值，这个值就称为矩阵所对应的行列式的值。MATLAB 中，求方阵 A 所对应的行列式的值的函数是 det(A)。

例 5.7　用克拉默(Cramer)方法求解线性方程组

$$\begin{cases} 2x_1+2x_2 - x_3 + x_4 = 4 \\ 4x_1+3x_2 - x_3 + 2x_4 = 6 \\ 8x_1+5x_2 - 3x_3 + 4x_4 = 12 \\ 3x_1+3x_2 - 2x_3 + 2x_4 = 6 \end{cases}$$

根据克拉默(Cramer)方法求解一个 n 阶的线性方程组要计算 $n+1$ 个行列式的值。在 MATLAB 中，要构造 $n+1$ 个矩阵，下面通过 MATLAB 提供的函数来完成该运算。程序如下：

```
D = [2,2,-1,1;4,3,-1,2;8,5,-3,4;3,3,-2,2];     %定义系数矩阵
b = [4;6;12;6];                                %定义常数项向量
D1 = [b,D(:,2:4)];                             %用方程组的右端向量置换 D 的第 1 列
D2 = [D(:,1:1),b,D(:,3:4)];                    %用方程组的右端向量置换 D 的第 2 列
D3 = [D(:,1:2),b,D(:,4:4)];                    %用方程组的右端向量置换 D 的第 3 列
D4 = [D(:,1:3),b];                             %用方程组的右端向量置换 D 的第 4 列
DD = det(D);
x1 = det(D1)/DD;
x2 = det(D2)/DD;
x3 = det(D3)/DD;
x4 = det(D4)/DD;
[x1,x2,x3,x4]
```

程序执行后的输出为：

```
ans =
     1     1    -1    -1
```

5.2.4 矩阵的秩

1. 向量的线性相关

为理解矩阵秩的概念，先对向量的线性相关及其相关概念进行概述。设 $\{x_1,x_2,\cdots,x_n\}$ 是 n 个向量，并且每个向量含有同样多个元素，$\{k_1,k_2,\cdots,k_n\}$ 是 n 个不全为 0 的常数。称 $x=k_1x_1+k_2x_2+\cdots+k_nx_n$ 为向量组 $\{x_1,x_2,\cdots,x_n\}$ 的线性组合，或者称为 $\{x_1,x_2,\cdots,x_n\}$ 的线性表示。

(1) 线性相关性

若存在不全为 0 的一组常数 $\{k_1,k_2,\cdots,k_n\}$，使得

$$k_1x_1+k_2x_2+\cdots+k_nx_n=\mathbf{0}$$

称向量组 $\{x_1,x_2,\cdots,x_n\}$ 线性相关，否则，称其线性无关。换句话说，当一组向量能线性组合成 $\mathbf{0}$ 向量时，称其为一组线性相关的向量，否则，称其为线性无关的向量。

(2) 线性无关组

线性无关的一组向量称为线性无关组。若向量组$\{x_1, x_2, \cdots, x_n\}$是满足某种条件的线性无关组，并且在该无关组中任意添加一个满足条件的向量 x，则向量组$\{x, x_1, x_2, \cdots, x_n\}$线性相关，称$\{x_1, x_2, \cdots, x_n\}$为最大线性无关组。对于三维向量，设 $x_1 = (1,0,0)$，$x_2 = (0,1,0)$，$x_3 = (0,0,1)$，显然，$\{x_1, x_2, x_3\}$组成所有三维向量的最大无关组。向量的线性无关组有一个基本性质：组成线性无关组的向量可以不同，但其个数是确定的。对于三维向量，可以找到不同于上述$\{x_1, x_2, x_3\}$的另一个最大线性无关组，但该最大无关组也一定是由 3 个线性无关的向量组成的。

2. 矩阵的秩

一个矩阵中行(列)向量的最大线性无关组包含的行(列)向量的个数称为该矩阵的秩。也可把矩阵的秩定义为矩阵的非零子式(子式就是一个矩阵的部分元素组成的行列式)的最大阶数，后一种定义更便于实际计算一个矩阵的秩。如果一个矩阵的所有行向量线性无关，称该矩阵为行满秩；一个矩阵的所有列向量线性无关，称该矩阵为列满秩。行(列)满秩的方阵称为满秩矩阵。MATLAB 中，求矩阵秩的函数是 rank(A)。例如，求例 5.7 中方程组系数矩阵 $\boldsymbol{D}$ 的秩，命令及运行结果如下：

```
D = [2,2,-1,1;4,3,-1,2;8,5,-3,4;3,3,-2,2];
r = rank(D)
r =
    4
```

说明：$\boldsymbol{D}$ 是一个满秩矩阵。

5.2.5 向量和矩阵的范数

向量和矩阵的范数是定义在向量和矩阵上的一种非负实值函数。设 $\boldsymbol{X}$ 是一个任意向量或矩阵，$\boldsymbol{X}$ 对应一个非负实数 $\|\boldsymbol{X}\|$，并且，$\|\boldsymbol{X}\|$ 应满足 3 种性质：

(1) 当 $\boldsymbol{X} \neq 0$时，$\|\boldsymbol{X}\| > 0$；$\boldsymbol{X} = 0$ 时，$\|\boldsymbol{X}\| = 0$。

(2) 对于任意常数 k，有 $\|k\boldsymbol{X}\| = |k| \|\boldsymbol{X}\|$。

(3) 对于任意两个向量或矩阵 $\boldsymbol{X}, \boldsymbol{Y}$，成立三角形不等式

$$\|\boldsymbol{X} + \boldsymbol{Y}\| \leqslant \|\boldsymbol{X}\| + \|\boldsymbol{Y}\|$$

则称这样的函数 $\|\boldsymbol{X}\|$ 为 $\boldsymbol{X}$ 的范数。这里不讨论范数的一般性质，但是，读者应明确，向量和矩阵都有多种方法定义其范数，范数的定义不同，范数值也就不同，因此，讨论向量和矩阵的范数时，一定要弄清是求哪一种范数。

1. 向量的 3 种常用范数及其计算函数

设向量 $\boldsymbol{V} = (v_1, v_2, \cdots, v_n)$，下面讨论向量的 3 种范数：

(1) 2 - 范数　$\|\boldsymbol{V}\|_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$

(2) 1 - 范数　$\|\boldsymbol{V}\|_1 = |v_1| + |v_2| + \cdots + |v_n|$

(3) ∞ - 范数　$\|\boldsymbol{V}\|_\infty = \max\{|v_i|\}$

从上述定义可以看到，2 - 范数就是向量的模。MATLAB 中，求这 3 种向量范数的函数为：

(1) norm(V)或 norm(V,2)　计算向量 $\boldsymbol{V}$ 的 2 - 范数

(2) norm(V,1)　计算向量 $\boldsymbol{V}$ 的 1 - 范数

(3) norm(V,inf)　计算向量 $\boldsymbol{V}$ 的 ∞ - 范数

例 5.8　设 $V=(-1,\frac{1}{2},1)$，求 V 的 3 种范数。

命令及运行结果如下：

```
V = [-1,1/2,1];
v1 = norm(V,1)              %求 V 的 1-范数
v1 =
    2.5000
v2 = norm(V)                %求 V 的 2-范数
v2 =
    1.5000
vinf = norm(V,inf)          %求 V 的∞-范数
vinf =
     1
```

本例清楚地说明了，同一个向量的不同范数值不一定相同。

2. 矩阵的范数及其计算函数

前面已经说明了向量的 3 种常用范数及其在 MATLAB 中的计算函数，下面以向量的范数为基础来定义矩阵的范数。

设 $\boldsymbol{A}$ 是一个 $m\times n$ 的矩阵，$\boldsymbol{V}$ 是一个含有 n 个元素的列向量，定义

$$\|\boldsymbol{A}\| = \max\|\boldsymbol{AV}\|,\ \|\boldsymbol{V}\| = 1$$

因为 $\boldsymbol{A}$ 是一个 $m\times n$ 的矩阵，而 $\boldsymbol{V}$ 是一个含有 n 个元素的列向量，所以 $\boldsymbol{A}\cdot\boldsymbol{V}$ 是一个含有 n 个元素的列向量。前面已经定义了 3 种不同的向量范数，按照上式也可定义 3 种矩阵范数，这样定义的矩阵范数 $\|\boldsymbol{A}\|$ 称为 $\boldsymbol{A}$ 从属于向量的范数。

上式只给出了矩阵范数的基本定义，未给出具体计算方法，完全按照上式是难于计算一个矩阵的某种具体范数的。从属于 3 种向量范数的矩阵范数计算公式是（a_{ik}系矩阵 $\boldsymbol{A}$ 的元素）：

(1) $\|\boldsymbol{A}\|_1 = \max\limits_{\|V\|=1}\{\|\boldsymbol{AV}\|_1\} = \max\limits_{k}\{\sum\limits_{i=1}^{m}|a_{ik}|\}$

(2) $\|\boldsymbol{A}\|_2 = \max\limits_{\|V\|=1}\{\|\boldsymbol{AV}\|_2\} = \sqrt{\lambda_1}$，其中 λ_1 为 $\boldsymbol{A}'\boldsymbol{A}$ 最大特征值。

(3) $\|\boldsymbol{A}\|_\infty = \max\limits_{\|V\|=1}\{\|\mathrm{AV}\|_\infty\} = \max\limits_{i}\{\sum\limits_{k=1}^{n}|a_{ik}|\}$

MATLAB 中提供了求 3 种矩阵范数的函数，其函数调用格式与求向量的范数的函数完全相同，这里不再赘述。

例 5.9　求矩阵 $\boldsymbol{A}$ 的 3 种范数。

命令及运行结果如下：

```
A = [17,0,1,0,15;23,5,7,14,16;4,0,13,0,22;10,12,19,21,3;11,18,25,2,19];
a1 = norm(A,1)              %求 A 的 1-范数
a1 =
    75
a2 = norm(A)                %求 A 的 2-范数
a2 =
    59.3617
```

```
ainf = norm(A,inf)              %求 A 的∞ - 范数
ainf =
     75
```

5.2.6 矩阵的条件数和迹

1. 矩阵的条件数

在求解线性方程组 $AX = b$ 时，一般认为，系数矩阵 A 中个别元素的微小扰动不会引起解向量的很大变化。这样的假设在工程应用中非常重要，因为一般系数矩阵的数据是由实验数据获得的，并非精确值，但与精确值误差不大，上面的假设可以得出如下结论：当参与运算的系数与实际精确值误差很小时，所获得的解与问题的准确解误差也很小。遗憾的是上述假设并非总是正确的。对于有的系数矩阵，个别元素的微小扰动会引起解的很大变化，在计算数学中，称这种矩阵是病态矩阵，而称解不因系数矩阵的微小扰动而发生大的变化的矩阵为良性矩阵。当然，良性与病态是相对的，需要一个参数来描述，条件数就是用来描述矩阵的这种性能的一个参数。定义

$$\mathrm{cond}(A) = \|A\| \, \|A^{-1}\|$$

即矩阵 A 的条件数等于 A 的范数与 A 的逆矩阵的范数的乘积。这样定义的条件数总是大于 1 的。条件数越接近于 1，矩阵的性能越好，反之，矩阵的性能越差。A 有 3 种范数，相应地可定义 3 种条件数。MATLAB 中，计算 A 的 3 种条件数的函数是：

(1) cond(A,1)　计算 A 的 1 - 范数下的条件数，即

$$\mathrm{cond}(A,1) = \|A\|_1 \|A^{-1}\|_1$$

(2) cond(A)或 cond(A,2)　计算 A 的 2 - 范数下的条件数，即

$$\mathrm{cond}(A) = \|A\|_2 \|A^{-1}\|_2$$

(3) cond(A,inf)　计算 A 的∞ - 范数下的条件数，即

$$\mathrm{cond}(A,\mathrm{inf}) = \|A\|_\infty \|A^{-1}\|_\infty$$

例 5.10　求矩阵

$$X = \begin{bmatrix} 2 & 2 & 3 \\ 4 & 5 & -6 \\ 7 & 8 & 9 \end{bmatrix}$$

的三种条件数。

命令及运行结果如下：

```
A = [2,2,3;4,5,-6;7,8,9];
C1 = cond(A,1)
C1 =
    149.1429
C2 = cond(A)
C2 =
    87.9754
C3 = cond(A,inf)
C3 =
    144.0000
```

2. 矩阵的迹

矩阵的迹等于矩阵的对角线元素之和,也等于矩阵的特征值之和。MATLAB中,求矩阵的迹的函数是trace(A)。例如

```
X=[2 2 3;4 5 -6;7 8 9];
trace(X)
ans =
     16
```

5.2.7 矩阵的特征值与特征向量

在物理学、力学和工程技术中遇到的机械振动、电磁振荡及其稳定性问题,往往可归结为如下的数学模型。对于n阶方阵$\boldsymbol{A}$,求数λ和向量ξ,使得等式$\boldsymbol{A}\xi=\lambda\xi$成立。满足等式的数$\lambda$称为$\boldsymbol{A}$的特征值,而向量$\xi$称为$\boldsymbol{A}$的特征向量。实际上,方程$\boldsymbol{A}\xi=\lambda\xi$和$(\boldsymbol{A}-\lambda\boldsymbol{I})\xi=\boldsymbol{0}$是两个等价方程。要使方程$(\boldsymbol{A}-\lambda\boldsymbol{I})\xi=\boldsymbol{0}$有非$\boldsymbol{0}$解$\xi$,必须使其系数行列式为0,即$\det(\boldsymbol{A}-\lambda\boldsymbol{I})=0$。

线性代数中已经证明,行列式$\det(\boldsymbol{A}-\lambda\boldsymbol{I})$是一个关于$\lambda$的$n$次多项式,因而方程$\det(\boldsymbol{A}-\lambda\boldsymbol{I})=0$是一个$n$次方程,有$n$个根(含重根),就是矩阵$\boldsymbol{A}$的$n$个特征值,每一个特征值对应无穷多个特征向量。矩阵的特征值问题有确定解,但特征向量问题没有确定解。

特征值和特征向量在科学研究和工程计算中都有非常广泛的应用。MATLAB中,计算矩阵$\boldsymbol{A}$的特征值和特征向量的函数是eig(A),常用的调用格式有3种:

(1) E=eig(A) 求矩阵$\boldsymbol{A}$的全部特征值,构成向量$\boldsymbol{E}$。

(2) [V,D]=eig(A) 求矩阵$\boldsymbol{A}$的全部特征值,构成对角阵$\boldsymbol{D}$,并求$\boldsymbol{A}$的特征向量构成$\boldsymbol{V}$的列向量。

(3) [V,D]=eig(A,'nobalance') 与第2种格式类似,但第2种格式中先对$\boldsymbol{A}$作相似变换,后求矩阵$\boldsymbol{A}$的特征值和特征向量,而格式3直接求矩阵$\boldsymbol{A}$的特征值和特征向量。

一个矩阵的特征向量有无穷多个,eig函数只找出其中的n个,$\boldsymbol{A}$的其他特征向量,均可由这n个特征向量的线性组合表示。

例5.11 设矩阵

$$\boldsymbol{A}=\begin{bmatrix}1 & 2 & 2\\1 & -1 & 1\\4 & -12 & 1\end{bmatrix}$$

用3种不同的格式求$\boldsymbol{A}$的特征值和特征向量。

命令及运行结果如下:

```
A=[1,2,2;1,-1,1;4,-12,1];
E=eig(A)
E =
   1.0000
   0.0000 + 1.0000i
   0.0000 - 1.0000i
[V,D]=eig(A)
V =
```

```
    0.9045              -0.4432 + 0.5744i   -0.4432 - 0.5744i
    0.3015              -0.1904 + 0.1280i   -0.1904 - 0.1280i
   -0.3015               0.1248 - 0.6368i    0.1248 + 0.6368i
D =
    1.0000                   0                   0
         0           0.0000 + 1.0000i            0
         0                   0           0.0000 - 1.0000i
[V,D] = eig(A,'nobalance')
V =
   -0.9045              0.2689 - 0.6738i   0.2689 + 0.6738i
   -0.3015              0.1480 - 0.1753i   0.1480 + 0.1753i
    0.3015              0.0544 + 0.6466i   0.0544 - 0.6466i
D =
    1.0000                   0                   0
         0           0.0000 + 1.0000i            0
         0                   0           0.0000 - 1.0000i
```

可以看出,用不同的调用格式求得的特征值都是(1,i,-i),但所求的特征向量并不完全相同。

例 5.12 用求特征值的方法解方程

$$3x^5 - 7x^4 + 5x^2 + 2x - 18 = 0$$

先构造与方程对应的多项式的友矩阵 $\boldsymbol{A}$,再求 $\boldsymbol{A}$ 的特征值。$\boldsymbol{A}$ 的特征值即为方程的根。命令及运行结果如下:

```
p = [3,-7,0,5,2,-18];
A = compan(p);                    %A 是多项式 p 的友矩阵
x1 = eig(A)                       %求 A 的特征值
x1 =
    2.1837
    1.0000 + 1.0000i
    1.0000 - 1.0000i
   -0.9252 + 0.7197i
   -0.9252 - 0.7197i
x2 = roots(p)                     %直接多项式 p 的零点
x2 =
    2.1837
    1.0000 + 1.0000i
    1.0000 - 1.0000i
   -0.9252 + 0.7197i
   -0.9252 - 0.7197i
```

可以看出,两种方法求得的方程的根是完全一致的,实际上,roots 函数正是应用求友矩阵的特征值的方法来求方程的根。

5.2.8 MATLAB 在三维向量中的应用

三维向量在解析几何和高等数学中均有广泛的应用，本节讨论如何利用 MATLAB 来实现三维向量的一些基本运算，所讨论的大部分问题，MATLAB 中并未直接提供相应的处理函数，需要使用命令序列或程序来完成。

1. 向量共线或共面的判断

从线性代数中知道，当两个向量线性相关时，则这两个向量组成的矩阵的秩小于 2，并且这两个向量共线，否则不共线。当 3 个向量线性相关时，则这 3 个向量组成的矩阵的秩小于 3，并且这 3 个向量共面，否则不共面。根据这些知识，很容易在 MATLAB 中判断三维向量的共线、共面问题。

例 5.13　设 $\boldsymbol{X}=(1,1,1)$，$\boldsymbol{Y}=(-1,2,1)$，$\boldsymbol{Z}=(2,2,2)$，判断这 3 个向量的共线、共面问题。

命令如下：

```
X=[1,1,1];Y=[-1,2,1];Z=[2,2,2];
XY=[X;Y];YZ=[Y;Z];ZX=[Z;X];XYZ=[X;Y;Z];
rank(XY)
rank(YZ)
rank(ZX)
rank(XYZ)
```

结果分别是 2,2,1,2。说明 $\boldsymbol{X}$、$\boldsymbol{Y}$ 不共线，$\boldsymbol{Y}$、$\boldsymbol{Z}$ 不共线，$\boldsymbol{Z}$、$\boldsymbol{X}$ 共线，$\boldsymbol{X}$、$\boldsymbol{Y}$、$\boldsymbol{Z}$ 共面（但不共线）。

作为练习，请读者根据上述方法设计一个函数文件，其参数是待判断的向量，函数值是判断结果。当 2 个向量共线或 3 个向量共面时，函数返回 0，否则返回 1。

2. 向量方向余弦的计算

设 $\boldsymbol{V}=(x,y,z)$ 是一个空间三维向量，r 是 $\boldsymbol{V}$ 的长度，称向量 $\boldsymbol{D}=(x/r,y/r,z/r)$ 是向量 $\boldsymbol{V}$ 的方向余弦。给出一个向量后，很容易求得该向量的长度 r=norm(V,2)，从而可以方便地求得其方向余弦。

例 5.14　设向量 $\boldsymbol{V}=(5,-3,2)$，求 $\boldsymbol{V}$ 的方向余弦。

建立一个函数文件 direct.m：

```
function f=f(v)
r=norm(v);
if r==0
    f=0
else
    f=[v(1)/r,v(2)/r,v(3)/r];
end
return
```

在 MATLAB 命令窗口，输入命令：

```
v=[5,-3,2];
f=direct(v)
f =
```

```
    0.8111    -0.4867    0.3244
```

当然,也可不设计函数文件,而在 MATLAB 命令窗口直接输入命令:

```
V=[5,-3,2];r=norm(V);
D=[V(1)/r,V(2)/r,V(3)/r]
D =
    0.8111    -0.4867    0.3244
```

3. 向量的夹角

设 $\boldsymbol{U}=(u_1,u_2,u_3)$, $\boldsymbol{V}=(v_1,v_2,v_3)$是2个空间向量,$r_1$、$r_2$分别是 $\boldsymbol{U}$、$\boldsymbol{V}$ 的长度,$(\boldsymbol{U},\boldsymbol{V})=u_1v_1+u_2v_2+u_3v_3$ 是 $\boldsymbol{U}$、$\boldsymbol{V}$ 的内积,θ 是 $\boldsymbol{U}$、$\boldsymbol{V}$ 的夹角,从解析几何中知道,向量的内积与夹角间的关系是

$$(\boldsymbol{U},\boldsymbol{V})=r_1r_2\cos\theta$$

给出两个向量后,很容易通过 MATLAB 求得其内积和向量的长度 r_1、r_2,从而通过上式可求得两个向量的夹角 θ。

例 5.15 设 $\boldsymbol{U}=(1,0,0)$, $\boldsymbol{V}=(0,1,0)$,求 $\boldsymbol{U}$, $\boldsymbol{V}$ 间的夹角 θ。

命令及运行结果如下:

```
U=[1,0,0];V=[0,1,0];
r1=norm(U);r2=norm(V);
UV=U*V';cosd=UV/r1/r2;
D=acos(cosd)
D =
    1.5708
```

结果为1.5708弧度,即90°。容易看出,$\boldsymbol{U}$、$\boldsymbol{V}$ 分别是直角坐标系中的两个坐标轴 Ox、Oy 上的单位向量,其夹角显然是90°。

4. 两点间的距离

设 $\boldsymbol{U}=(u_1,u_2,u_3)$, $\boldsymbol{V}=(v_1,v_2,v_3)$,这里,$\boldsymbol{U}$、$\boldsymbol{V}$ 即可以理解为两个空间点的坐标,也可以理解为从坐标原点到该点的向量。当对其理解为两个点坐标时,一个很自然的问题就是怎样求这两个点之间的距离。从解析几何中知,这两个向量的差向量的长度即为两点间的距离。

例 5.16 设 $\boldsymbol{U}=(1,0,0)$, $\boldsymbol{V}=(0,1,0)$,求 $\boldsymbol{U}$、$\boldsymbol{V}$ 两点间的距离。

命令如下:

```
U=[1,0,0];V=[0,1,0];UV=U-V;
D=norm(UV)
D =
    1.4142
```

5. 向量的向量积

向量 $\boldsymbol{U}$、$\boldsymbol{V}$ 的向量积也是一个向量,记为 $\boldsymbol{U}\times\boldsymbol{V}$。$\boldsymbol{U}\times\boldsymbol{V}$ 的方向与 $\boldsymbol{U}$、$\boldsymbol{V}$ 组成的平而垂直,$\boldsymbol{U}$、$\boldsymbol{V}$、$\boldsymbol{U}\times\boldsymbol{V}$ 3个向量间遵守右手系;$\boldsymbol{U}\times\boldsymbol{V}$ 的长度等于 $\boldsymbol{U}$ 的长度与 $\boldsymbol{V}$ 的长度及 $\boldsymbol{U}$、$\boldsymbol{V}$ 间的夹角的正弦三者的乘积,在数值上等于以 $\boldsymbol{U}$、$\boldsymbol{V}$ 为邻边组成的平行四边形的面积。向量积在求与已知平面垂直的向量及两个已知向量组成的平行四边形而积时非常有用。

设 $\boldsymbol{U}=(u_1,u_2,u_3)$, $\boldsymbol{V}=(v_1,v_2,v_3)$,根据向量代数的知识,$\boldsymbol{U}$、$\boldsymbol{V}$ 的向量积 $\boldsymbol{U}\times\boldsymbol{V}$ 可以用一

个行列式来表达：

$$U \times V = \begin{vmatrix} i & j & k \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}$$

式中 i、j、k 分别是坐标轴 Ox、Oy、Oz 的单位向量。MATLAB 没有直接提供求两个向量的向量积的函数，但利用上述行列式可以计算向量的向量积。

例 5.17 设 $U=(2,-3,1)$，$V=(3,0,4)$，求 $U \times V$。

命令如下：

```
U=[2,-3,1];V=[3,0,4];W=eye(3);
A1=[W(1,:);U;V];A2=[W(2,:);U;V];A3=[W(3,:);U;V];
UV=[det(A1),det(A2),det(A3)]
UV =
    -12     -5     9
```

6. 向量的混合积

设 $U=(u_1,u_2,u_3)$，$V=(v_1,v_2,v_3)$，$W=(w_1,w_2,w_3)$，U、V、W 的混合积定义为 $(UVW)=U(V \times W)$，向量代数中已经得出，3 个向量的混合积可以用一个行列式来表达，即

$$U(V \times W) = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}$$

利用上述行列式就可以写出计算 3 个向量的混合积的 MATLAB 语句了。

例 5.18 设 $U=(0,0,2)$，$V=(3,0,5)$，$W=(1,1,0)$，求以这 3 个向量构成的六面体的体积。

这 3 个向量的混合积即是其构成的六面体的体积。命令及运行结果如下：

```
U=[0,0,2];V=[3,0,5];W=[1,1,0];
A=[U;V;W];
det(A)
ans =
    6
```

7. 点到平面的距离

从解析几何知，点 $U=(u_1,u_2,u_3)$ 到平面 $Ax+By+Cz+D=0$ 的距离 r 的计算公式是

$$r=\frac{|Au_1+Bu_2+Cu_3+D|}{\sqrt{A^2+B^2+C^2}}$$

根据上式在 MATLAB 中就能方便地计算点到平面的距离。

例 5.19 求原点到平面 $x+y+z=1$ 的距离。

命令及运行结果如下：

```
u=[0,0,0];v=[1,1,1];          % A=B=C=1,u1=u2=u3=0,D=-1
r=abs(u*v'-1)/norm(v,2)
r =
   0.5774
```

5.3 矩阵分解与线性方程组求解

5.3.1 矩阵分解

与初等代数中因式分解过程一样,线性代数中将一个矩阵表示为几个矩阵的乘积的过程称为矩阵分解。矩阵分解在很多矩阵运算过程中都有应用。MATLAB 中有许多函数可进行矩阵分解,下面讨论其中几个常用的分解函数。

1. 实对称矩阵的 QDQ 分解

如果一个矩阵的转置矩阵和自身相等,则称该矩阵为对称矩阵,进一步,如果对称矩阵的所有元素均为实数,则称该矩阵为实对称矩阵。从线性代数中知道,对任何一个实对称矩阵 $\boldsymbol{A}$,都存在正交矩阵 Q(正交矩阵是满足正交条件 $Q'Q = QQ' = \boldsymbol{I}$ 的矩阵),使得

$$Q'\boldsymbol{A}Q = \boldsymbol{D} \qquad (\boldsymbol{D}\text{ 为对角矩阵})$$

从而有

$$\boldsymbol{A} = Q\boldsymbol{D}Q'$$

这里,$\boldsymbol{D}$ 是由 $\boldsymbol{A}$ 的特征值组成的对角矩阵,而 Q 是对应的特征向量。特征值和特征向量都可以通过 eig(A)求得,因此,在 MATLAB 中,可以方便地将实对称矩阵 $\boldsymbol{A}$ 分解为 3 个矩阵乘积的形式。

例 5.20 设对称矩阵

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 & 4 & 6 \\ 1 & 2 & 1 & 5 \\ 4 & 1 & 3 & 4 \\ 6 & 5 & 4 & 2 \end{bmatrix}$$

对 $\boldsymbol{A}$ 进行 QDQ 分解。

命令如下:

```
A = [2,1,4,6;1,2,1,5;4,1,3,4;6,5,4,2];
[Q,D] = eig(A)
Q =
   -0.6081    0.2983    0.5157  0.5247
     0.2885   -0.7789    0.4260  0.3586
     0.7113    0.5139    0.0570  0.4762
   -0.2026   -0.2006   -0.7412  0.6077
D =
   -1.1538              0              0              0
          0         2.2449              0              0
          0              0        -5.3556              0
          0              0              0        13.2645
Q * D * Q'
```

```
ans =
    2.0000    1.0000    4.0000    6.0000
    1.0000    2.0000    1.0000    5.0000
    4.0000    1.0000    3.0000    4.0000
    6.0000    5.0000    4.0000    2.0000
```

结果与 ***A*** 相等，说明确实将 ***A*** 分解为了 QDQ′的乘积。

例 5.21 求下列二次型的标准形式及变换矩阵。

$$f(x,y,z)=x^2+y^2+3z^2+4xy+2yz+2zx$$

设

$$\boldsymbol{A}=\begin{bmatrix}1&2&1\\2&1&1\\1&1&3\end{bmatrix}\qquad \boldsymbol{X}=\begin{bmatrix}x\\y\\z\end{bmatrix}$$

则有 $f(x,y,z)=\boldsymbol{X}'\boldsymbol{A}\boldsymbol{X}$。

这里 ***A*** 是一个实对称矩阵，可进行 QDQ 分解，即存在正交矩阵 ***Q*** 和对角矩阵 ***D***，使 $\boldsymbol{A}=\boldsymbol{QDQ}'$。从而有 $f(x,y,z)=\boldsymbol{X}'(\boldsymbol{QDQ}')\boldsymbol{X}=(\boldsymbol{X}'\boldsymbol{Q})\boldsymbol{D}(\boldsymbol{Q}'\boldsymbol{X})=(\boldsymbol{Q}'\boldsymbol{X})'\boldsymbol{D}(\boldsymbol{Q}'\boldsymbol{X})$。作线性变换 $\boldsymbol{Y}=\boldsymbol{Q}'\boldsymbol{X}$，则原二次型变换为二次标准型。命令如下：

```
A=[1,2,1;2,1,1;1,1,3;];
[Q,D]=eig(A)
Q =
    0.7071   -0.5000   0.5000
   -0.7071   -0.5000   0.5000
    0.0000    0.7071   0.7071
D =
   -1.0000         0         0
         0    1.5858         0
         0         0    4.4142
```

这里 $\boldsymbol{A}=\boldsymbol{QDQ}'$，作线性变换：$[u,v,w]'=\boldsymbol{Q}'\boldsymbol{X}=\boldsymbol{Q}'[x,y,z]'$，即

$$\begin{cases}u=0.707\,1x-0.707\,1y\\v=-0.5x-0.5y+0.707\,1z\\w=0.5x+0.5y+0.707\,1z\end{cases}$$

其逆变换是：

$$\begin{cases}x=0.707\,1u-0.5v+0.5w\\y=-0.707\,1u-0.5v+0.5w\\z=0.707\,1v+0.707\,1w\end{cases}$$

原关于 x,y,z 的二次型化为关于 u,v,w 的标准二次型

$$f(x,y,z)=g(u,v,w)=-u^2+1.585\,8v^2+4.414\,2w^2$$

2. 矩阵的 LU 分解

矩阵的 LU 分解就是将一个矩阵表示一个可交换下三角矩阵和一个上三角矩阵的乘积形式。线性代数中已经证明，只要方阵 ***A*** 是非奇异的，即 $\det(A)\neq 0$ 时，LU 分解总是可以进行的。

MATLAB 中,完成 LU 分解的函数是:

(1) [L,U] = lu(A)　将方阵 $\boldsymbol{A}$ 分解为可交换下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$,使 $\boldsymbol{A}=\boldsymbol{LU}$。

(2) [L,U,P] = lu(A)　将方阵 $\boldsymbol{A}$ 分解为下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$,使 $\boldsymbol{PA}=\boldsymbol{LU}$。

当使用第 1 种格式时,矩阵 $\boldsymbol{L}$ 往往不是一个下三角矩阵,但可以通过行交换成为一个下三角阵。设

$$\boldsymbol{A}=\begin{bmatrix}2 & 1 & -2\\ 1 & 2 & 1\\ 2 & 5 & 3\end{bmatrix}$$

则对矩阵 $\boldsymbol{A}$ 进行 LU 分解的命令如下:

```
A = [2,1,-2;1,2,1;2,5,3];
[L,U] = lu(A)
L =
    1.0000         0         0
    0.5000    0.3750    1.0000
    1.0000    1.0000         0
U =
    2.0000    1.0000   -2.0000
         0    4.0000    5.0000
         0         0    0.1250
```

为检验结果是否正确,输入命令:

```
LU = L * U
LU =
     2     1    -2
     1     2     1
     2     5     3
```

说明结果是正确的。例中所获得的矩阵 $\boldsymbol{L}$ 并不是一个下三角矩阵,但将第二、三行互换后,即可获得一个下三角矩阵。

下面通过一个实例介绍应用 LU 分解求解线性方程组的方法。

例 5.22　用 LU 分解求方程组

$$\begin{cases}2x_1+5x_2+4x_3+x_4=20\\ x_1+3x_2+2x_3+x_4=11\\ 2x_1+10x_2+9x_3+7x_4=40\\ 3x_1+8x_2+9x_3+2x_4=37\end{cases}$$

的根。

写出方程组的系数矩阵 $\boldsymbol{A}$,右端项向量 $\boldsymbol{b}$,并调用 lu(A)对 $\boldsymbol{A}$ 进行分解,得

$$\boldsymbol{A}=\begin{bmatrix}2 & 5 & 4 & 1\\ 1 & 3 & 2 & 1\\ 2 & 10 & 9 & 7\\ 3 & 8 & 9 & 2\end{bmatrix},\quad \boldsymbol{X}=\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix}20\\ 11\\ 40\\ 37\end{bmatrix},\quad \boldsymbol{P}=\begin{bmatrix}0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}$$

原方程组为

$$AX = b$$

以 LU 分解结果 $PA = LU$ 代入，得

$$(LU)X = Pb$$

即：

$$L(UX) = Pb$$

作变换 $Y = UX$，得

$$LY = Pb$$

即原方程组的求解转化为对 2 个三角形方程组的求解：

$$\begin{cases} y_1 = 37 \\ 0.6667y_1 + y_2 = 40 \\ 0.6667y_1 - 0.0714y_2 + y_3 = 2 \\ 0.3333y_1 + 0.0714y_2 + 0.68y_3 + y_4 = 10 \end{cases}$$

这个方程组很容易求解，从第 1 个方程开始求解，顺次将求解结果代入后面的方程，即可求得方程组的全部解。将求解结果代入下面的方程组，可求得原方程组 $AX = b$ 的解。

$$\begin{cases} 3x_1 + 8x_2 + 9x_3 + 2x_4 = y_1 \\ 0.6667x_2 + 3x_3 + 5.6667x_4 = y_2 \\ -1.7857x_3 + 0.0714x_4 = y_3 \\ -0.12x_4 = y_4 \end{cases}$$

和前面的方程组类似，这个方程组从第 4 个方程开始求解，顺次将求解结果代入前面的方程，可求得方程组的全部解。实际上，这就是高斯消元法的思想。本例的具体求解过程留作练习。

3. 矩阵的 QR 分解

对矩阵 A 进行 QR 分解，就是把 A 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的乘积形式。QR 分解只能对方阵进行。对矩阵 A 进行 QR 分解的函数是[Q,R] = qr(A)，根据方阵 A，求一个正交矩阵 Q 和一个上三角矩阵 R，使 $A = QR$。例如，对矩阵

$$A = \begin{bmatrix} 2 & 1 & -2 \\ 1 & 2 & 1 \\ 2 & 5 & 3 \end{bmatrix}$$

进行 QR 分解的命令是：

```
A = [2,1,-2;1,2,1;2,5,3];
[Q,R] = qr(A)
Q =
    -0.6667     0.7362     0.1162
    -0.3333    -0.1550    -0.9300
    -0.6667    -0.6587     0.3487
R =
    -3.0000    -4.6667    -1.0000
```

```
   0      -2.8674    -3.6037
   0         0       -0.1162
```

读者可以验证，上述结果 Q,R 满足 $Q'Q=QQ'=I,QR=A$，说明所得结果是正确的。

5.3.2 线性方程组求解

1. 线性方程组解的一般讨论

线性代数中对线性方程组 $Ax=b$ 的求解理论有详细讨论，这里对其作一个归纳：

(1) 当 A 是一个满秩方阵时，方程 $Ax=b$ 称为恰定方程，方程有惟一解 $x=A^{-1}b$，这是最基本的一种情况。一般用 x = A \ b 求解速度更快。

(2) 当方程组右端向量 $b=0$时，方程称为齐次方程组。齐次方程组总有零解，因此称解 $x=0$为平凡解。当 A 的秩小于n 时，齐次方程组有无穷多个非平凡解，其通解中包含 n - rank(A)个线性无关的解向量，用 MATLAB 的函数 null(A,'r')可求得基础解系。

(3) 一般而言，当右端向量 $b\neq0$时，方程组系数矩阵秩 rank(A)与其增广矩阵的秩 rank([A,b])是判断其是否有解的基本条件：

① 当 rank(A) = rank([A,b]) = n 时，方程组有惟一解：x = A \ b 或 x = pinv(A) * b。

② 当 rank(A) = rank([A,b]) < n 时，方程组有无穷多个解，其通解由一个特解和对应的齐次方程组 $Ax=0$的通解组成。可以用 A \ b 求得方程组的一个特解，用 null(A,'r')求得该方程组所对应的齐次方程组的基础解系，基础解系中包含 n - rank(A)个线性无关的解向量。

③ 当 rank(A) < rank([A,b])时，方程组无解，即没有 x，使得 $Ax=b$。应用向量范数理论，即没有 x，使得

$$\| Ax-b \|_2=0$$

由此想到，是否有 x，使得 $\| Ax-b \|_2$ 取最小值，这个最小值问题是有解的，其解为 x = A \ b或 x = pinv(A) * b，即 A 的伪逆矩阵与 b 的乘积。

有了上面这些讨论，就可以设计解线性方程组的一般函数了。函数文件如下：

```
function [x,y] = line_solution(A,b)
[m,n] = size(A);y = [];
if norm(b) > 0                    %非齐次方程组
   if rank(A) == rank([a,b])      %方程组相容
      if rank(A) == m             %有惟一解
         x = A \ b;
      else                        %方程组有无穷多个解,基础解系
         disp('原方程组有无穷个解,其齐次方程组的基础解系为 y,特解为 x');
         y = null(A,'r');
         x = A \ b;
      end
   else                           %方程组不相容,给出最小二乘法解
      disp('方程组的最小二乘法解是:');
      x = A \ b;
   end
```

```
else                          %齐次方程组
    if rank(A) >= n           %列满秩
        x = zero(m,1)         %0 解
    else                      %非 0 解
        disp('方程组有无穷个解,基础解系为 x');
        x = null(A,'r');
    end
end
return
```

2. 应用举例

例 5.23　求下列线性方程组

$$\begin{cases} 2x_1 + 2x_2 - x_3 + x_4 & = 4 \\ 4x_1 + 3x_2 - x_3 + 2x_4 & = 6 \\ 8x_1 + 5x_2 - 3x_3 + 4x_4 & = 12 \\ 3x_1 + 3x_2 - 2x_3 + 2x_4 & = 6 \end{cases}$$

的解。

在 MATLAB 命令窗口,输入命令:

```
A = [2,2,-1,1;4,3,-1,2;8,5,-3,4;3,3,-2,2];b = [4,6,12,6]';
[x,y] = line_solution(A,b)                %调用自定义函数
x =
    1.00000000000000
    1.00000000000000
   -1.00000000000000
   -1.00000000000000
y =
   []
```

结果 y = [],说明方程组只有惟一解。为验证结果,在 MATLAB 命令窗口,输入命令:

```
A * x
ans =
     4
     6
     12
     6
```

说明所求结果满足原方程组,是正确的。

例 5.24　求下列线性方程组

$$\begin{cases} 2x_1 + 7x_2 + 3x_3 + x_4 & = 6 \\ 3x_1 + 5x_2 + 2x_3 + 2x_4 & = 4 \\ 9x_1 + 4x_2 + x_3 + 7x_4 & = 2 \end{cases}$$

的解。

方程组有4个未知数，但只有3个方程，所以是欠定方程组，在其解存在的情况下，会有多个解。在MATLAB命令窗口，输入命令：

```
A=[2,7,3,1;3,5,2,2;9,4,1,7];b=[6,4,2]';
[x,y]=line_solution(A,b)
```

输出结果是：

```
原方程组有无穷个解，其齐次方程组的基础解系为y，特解为x
Warning: Rank deficient, rank = 2 tol = 8.6112e-015.
> In E:\MATLAB55\exam\line_solution.m at line 10
x =
  -0.18181818181818
   0.90909090909091
                  0
                  0
y =
   0.09090909090909   -0.81818181818182
  -0.45454545454545    0.09090909090909
   1.00000000000000                   0
                  0    1.00000000000000
```

y是原方程组所对应的齐次方程组的基础解系，包含2个线性无关的解向量。为检验解的正确性，输入命令：

```
A*x
ans =
   6.00000000000000
   4.00000000000000
   2.00000000000000
```

说明x确实是原方程组的一个特解。输入命令：

```
A*y
ans =
  1.0e-015 *
                  0                   0
   0.22204460492503                   0
   0.11102230246252   -0.88817841970013
```

结果很小，说明y是原方程组所对应的齐次方程组的解。理论上，A*y的结果应该是0向量，但由于计算误差，这里显示的结果已经很好了。

5.4 数据处理与多项式计算

5.4.1 数据统计与分析

1. 求矩阵最大和最小元素

(1) 求向量的最大最小元素

求一个向量 X 的最大元素的函数有两种调用格式，分别是：

① y = max(X) 返回向量 X 的最大元素存入 y，如果 X 中包含复数元素，则按模取最大元素。

② [y,I] = max(X) 返回向量 X 的最大元素存入 y，最大元素的序号存入 I，如果 X 中包含复数元素，则按模取最大元素。

求向量 X 的最小元素的函数是 min(X)，用法和 max(X)完全相同。

(2) 求矩阵的最大和最小元素

求矩阵 A 的最大元素的函数有 3 种调用格式，分别是：

① max(A) 返回一个行向量，向量的第 i 个元素是 A 矩阵的第 i 列上的最大元素。

② [Y,U] = max(A) 返回 2 个行向量，Y 向量记录 A 的每列的最大元素，U 向量记录每列最大元素的行号。

③ max(A,[],dim) *dim* 取 1 或 2。*dim* 取 1 时，该函数和 max(A)完全相同。*dim* 取 2 时，该函数返回一个列向量，其第 i 个元素是 A 矩阵的第 i 行上的最大元素。

求最小元素的函数是 min，其用法和 max 完全相同。

(3) 两个向量或矩阵对应元素的比较

函数 max 和 min 还能对两个同型的向量或矩阵进行比较：

① U = max(A,B) $\boldsymbol{A}$，$\boldsymbol{B}$ 是两个同型的向量或矩阵。结果 $\boldsymbol{U}$ 是与 $\boldsymbol{A}$，$\boldsymbol{B}$ 同型的向量或矩阵，$\boldsymbol{U}$ 的每个元素等于 $\boldsymbol{A}$，$\boldsymbol{B}$ 对应元素的较大者。

② U = max(A,n) n 是一个标量。结果 $\boldsymbol{U}$ 是与 $\boldsymbol{A}$ 同型的向量或矩阵，$\boldsymbol{U}$ 的每个元素等于 $\boldsymbol{A}$ 对应元素和 n 中的较大者。

min 函数的用法和 max 完全相同。

例 5.25 求矩阵

$$\boldsymbol{A}=\begin{bmatrix}13 & -56 & 78\\ 25 & 63 & -235\\ 78 & 25 & 563\\ 1 & 0 & -1\end{bmatrix}$$

的每行及每列的最大和最小元素，并求整个矩阵的最大和最小元。

命令如下：

```
A=[13,-56,78;25,63,-235;78,25,563;1,0,-1];
max(A,[],2)                          %求每行最大元素
```

```
ans =
    78
    63
   563
     1
min(A,[],2)                                    %求每行最小元素
ans =
   -56
  -235
    25
    -1
max(A)                                         %求每列最大元素
ans =
    78        63        563
min(A)                                         %求每列最小元素
ans =
     1       -56       -235
max(max(A))                                    %求整个矩阵的最大元素
ans =
   563
min(min(A))                                    %求整个矩阵的最小元素
ans =
  -235
```

2. 求矩阵的平均值和中值

所谓中值，是指在数据序列中其值的大小恰好处在中间的元素。例如，数据序列 -2,5,7,9,12 的中值为 7，因为比它大和比它小的数据均为 2 个，即它的大小恰好处于数据序列各个值的中间，这是数据序列为奇数个的情况；如果为偶数个，则中值等于中间的两项之平均值。例如，数据序列 -2,5,6,7,9,12 中，处于中间的数是 6 和 7，故其中值为此两数之平均值 6.5。

求矩阵和向量元素的平均值的函数是 mean，求中值的函数是 median。它们的调用方法和 max 函数完全相同。设 X 是一个向量，A 是一个矩阵，两个函数的用法如下：

mean(X)　返回向量 X 的算术平均值。

median(X)　返回向量 X 的中值。

mean(A)　返回一个行向量，其第 i 个元素是 A 的第 i 列的算术平均值。

median(A)　返回一个行向量，其第 i 个元素是 A 的第 i 列的中值。

mean(A,dim)　当 dim 为 1 时，该函数等同于 mean(A)，当 dim 为 2 时，返回一个列向量，其第 i 个元素是 A 的第 i 行的算术平均值。

median(A,dim)　当 dim 为 1 时，该函数等同于 median(A)，当 dim 为 2 时，返回一个列向量，其第 i 个元素是 A 的第 i 行的中值。

3. 矩阵元素求和与求积

矩阵和向量求和与求积的基本函数是 sum 和 prod，其使用方法和 max 类似，说明如下：

sum(X)　返回向量 $\boldsymbol{X}$ 各元素的和。

prod(X)　返回向量 $\boldsymbol{X}$ 各元素的乘积。

sum(A)　返回一个行向量,其第 i 个元素是 $\boldsymbol{A}$ 的第 i 列的元素和。

prod(A)　返回一个行向量,其第 i 个元素是 $\boldsymbol{A}$ 的第 i 列的元素乘积。

sum(A,dim)　当 *dim* 为 1 时,该函数等同于 sum(A),当 *dim* 为 2 时,返回一个列向量,其第 i 个元素是 $\boldsymbol{A}$ 的第 i 行的各元素和。

prod(A,dim)　当 *dim* 为 1 时,该函数等同于 prod(A),当 *dim* 为 2 时,返回一个列向量,其第 i 个元素是 $\boldsymbol{A}$ 的第 i 行的各元素乘积。

例 5.26　求矩阵 $\boldsymbol{A}$ 的每行元素的乘积和全部元素的乘积。

命令如下:

```
A=[1,2,3,4;5,6,7,8;9,10,11,12];
S=prod(A,2)
S =
        24
      1680
     11880
prod(S)                    %求 A 的全部元素的乘积
ans =
   479001600
```

4. 矩阵元素累加和与累乘积

设 $\boldsymbol{U}=(u_1,u_2,u_3,\cdots,u_n)$是一个向量,$\boldsymbol{V}$、$\boldsymbol{W}$ 是与 $\boldsymbol{U}$ 等长的另外两个向量,并且

$$\boldsymbol{V}=\left(\sum_{i=1}^{1}u_i,\sum_{i=1}^{2}u_i,\cdots,\sum_{i=1}^{n}u_i\right)$$

$$\boldsymbol{W}=\left(\prod_{i=1}^{1}u_i,\prod_{i=1}^{2}u_i,\cdots,\prod_{i=1}^{n}u_i\right)$$

称 $\boldsymbol{V}$ 为 $\boldsymbol{U}$ 的累加和向量,$\boldsymbol{W}$ 为 $\boldsymbol{U}$ 的累乘积向量。MATLAB 中,使用 cumsum 和 cumprod 函数能方便地求得向量和矩阵元素的累加和与累乘积向量,函数的用法和 sum 及 prod 相同,说明如下:

cumsum(X)　返回向量 $\boldsymbol{X}$ 累加和向量。

cumprod(X)　返回向量 $\boldsymbol{X}$ 累乘积向量。

cumsum(A)　返回一个矩阵,其第 i 列是 $\boldsymbol{A}$ 的第 i 列的累加和向量。

cumprod(A)　返回一个矩阵,其第 i 列是 $\boldsymbol{A}$ 的第 i 列的累乘积向量。

cumsum(A,dim)　当 *dim* 为 1 时,该函数等同于 cumsum(A),当 *dim* 为 2 时,返回一个矩阵,其第 i 行是 $\boldsymbol{A}$ 的第 i 行的累加和向量。

cumprod(A,dim)　当 *dim* 为 1 时,该函数等同于 cumprod(A),当 *dim* 为 2 时,返回一个向量,其第 i 行是 $\boldsymbol{A}$ 的第 i 行的累乘积向量。

例 5.27　求向量 $X=(1!,2!,3!,\cdots,10!)$。

命令如下:

```
X=cumprod(1:10)
X =
```

```
Columns 1 through 6
         1          2          6         24        120        720
Columns 7 through 10
      5040      40320     362880    3628800
```

5. 标准方差

对于具有 N 个元素的数据序列 $x_1, x_2, x_3, \cdots, x_N$，标准方差的计算公式如下

$$S_1 = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

或

$$S_2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

其中，$\bar{x}$ 为

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

标准方差的计算在数理统计中是一项基本工作。MATLAB 中，提供了计算数据序列的标准方差的函数 std。对于向量 $\boldsymbol{X}$，std(X)返回一个标准方差。对于矩阵 $\boldsymbol{A}$，std(A)返回一个行向量，它的各个元素便是矩阵 $\boldsymbol{A}$ 各列或各行的标准方差。std 函数的一般调用格式为：

```
std(A,FLAG,dim)
```

其中 dim 取 1 或 2。当 $dim = 1$ 时，求各列元素的标准方差；当 $dim = 2$ 时，则求各行元素的标准方差。FLAG 取 0 或 1，当 FLAG = 0 时，按 S_1 所列公式计算 S，当 FLAG = 1 时，按 S_2 所列公式计算 S。缺省 FLAG = 0，$dim = 1$。

6. 元素排序

对向量元素进行排序是一个经常性的操作，MATLAB 中对向量 $\boldsymbol{X}$ 是排序函数是 sort(X)，函数返回一个对 $\boldsymbol{X}$ 中的元素按升序排列的新向量。

sort 函数也可以对矩阵 $\boldsymbol{A}$ 的各列(或行)重新排序，其调用格式为：

```
[Y,I] = sort(A,dim)
```

其中 dim 指明对 $\boldsymbol{A}$ 的列还是行进行排序，若 $dim = 1$，则按列排，若 $dim = 2$，则按行排。$\boldsymbol{Y}$ 是排序后的矩阵，而 $\boldsymbol{I}$ 记录 $\boldsymbol{Y}$ 中的元素在 $\boldsymbol{A}$ 中的位置。

例 5.28 对下列矩阵做各种排序。

$$\boldsymbol{A} = \begin{bmatrix} 1 & -8 & 5 \\ 4 & 12 & 6 \\ 13 & 7 & -13 \end{bmatrix}$$

命令如下：

```
A = [1, -8,5;4,12,6;13,7, -13];
sort(A)                                    %对 A 的每列按升序排序
ans =
     1    -8   -13
     4     7     5
    13    12     6
```

```
- sort( - A,2)                          %对 A 的每行按降序排序
ans =
     5   1   -8
    12   6    4
    13   7  -13
[X,I] = sort(A)                         %对 A 按列排序,并将每个元素所在行号送矩阵 I
X =
     1   -8   -13
     4    7     5
    13   12     6
I =
     1    1    3
     2    3    1
     3    2    2
```

5.4.2 数值插值

设函数 $y=f(x)$,如果我们已经知道了对于自变量 X 的一组取值 $X=(x_1,x_2,\cdots,x_n)$,对应的函数值为

$$\begin{cases} y_1=f(x_1) \\ y_2=f(x_2) \\ \vdots \\ y_n=f(x_n) \end{cases}$$

称 $X=(x_1,x_2,\cdots,x_n)$为采样点,而 $Y=(y_1,y_2,\cdots,y_n)$是函数在采样点的样本值。数值插值的任务就是根据上述条件构造一个函数 $y=g(x)$,使得在 $X=(x_1,x_2,\cdots,x_n)$,有

$$\begin{cases} y_1=g(x_1) \\ y_2=g(x_2) \\ \vdots \\ y_n=g(x_n) \end{cases}$$

且在两个相邻的采样点$(x_i,x_{i+1})(i=1,2,\cdots,i-1)$之间,$g(x)$光滑过度。如果被插值函数 $f(x)$ 是光滑的,并且采样点足够密,一般在采样区间内,$f(x)$与 $g(x)$比较接近。插值函数 $g(x)$一般由线性函数、多项式、样条函数或这些函数的分段函数充当。

数值插值方法在实际工作中有广泛的应用。在工程测量和科学实验中,所得到的数据通常都是离散的。如果要得到这些离散点以外的其他点的数值,就需要根据这些已知数据进行插值。

根据被插值函数的自变量个数,插值问题分为一维插值,二维插值和多维插值等;根据是用分段直线、多项式或样条函数来作为插值函数,插值问题又分为线性插值、多项式插值和样条插值等。MATLAB 中,对上述问题均提供了相应的函数,下面逐一对其进行介绍。

1. 一维数值插值

如果被插值函数是一个单变量函数,则数值插值问题称为一维插值。一维插值采用的方法有线性方法、最近方法、三次样条和三次插值。MATLAB 中,实现这些插值的函数是 interp1。其

调用格式为：

```
Y1 = interp1(X,Y,X1,' method ')
```

函数根据 X、Y 的值，计算函数在 $X1$ 处的值。X、Y 是两个等长的已知向量，分别描述采样点和样本值，$X1$ 是一个向量或标量，描述欲插值的点，$Y1$ 是一个与 $X1$ 等长的插值结果。method 是插值方法，允许的取值有' linear '（线性插值）、' nearest '（最近插值）、' spline '（三次样条插值）、' cubic '（三次多项式插值），缺省值是' linear '。

注意：$X1$ 的取值范围不能超出 X 的给定范围，否则，会给出“NaN”错误。

例 5.29 用不同的插值方法计算 sin(x)在 $\pi/2$ 点的值。

这是一个一维插值问题。在 MATLAB 命令窗口，输入命令：

```
X = 0:0.2:pi;Y = sin(X);              %给出 X、Y
interp1(X,Y,pi/2)                     %用缺省方法(即线性插值方法)计算 sin(π/2)
ans =
    0.9975
interp1(X,Y,pi/2,' nearest ')         %用最近方法计算 sin(π/2)
ans =
    0.9996
interp1(X,Y,pi/2,' linear ')          %用线性方法计算 sin(π/2)
ans =
    0.9975
interp1(X,Y,pi/2,' spline ')          %用三次样条方法计算 sin(π/2)
ans =
    1.0000
interp1(X,Y,pi/2,' cubic ')           %用三次多项式方法计算 sin(π/2)
ans =
    1.0000
```

例中，三次样条和三次多项式的插值结果最好，最近方法次之，线性方法最差，但不能认为什么情况下都是这样的。插值方法的好坏依赖于被插值函数，没有一种对所有函数都是最好的插值方法。

MATLAB 中有一个专门的三次样条插值函数 Y1 = spline(X,Y,X1)，其功能及使用方法与函数 Y1 = interp1(X,Y,X1,' spline ')完全相同。

例 5.30 某检测参数 f 随时间 t 的采样结果如表 5.1，用数值插值法计算 t = 2,7,12,17,22,17,32,37,42,47,52,57 时 f 的值。

表 5.1 检测参数 f 与时间 t 的采样结果

t	0	5	10	15	20	25	30
f	3.2015	2.256	879.5	1835.9	2968.8	4136.2	5237.9
t	35	40	45	50	55	60	65
f	6152.7	6725.3	6848.3	6403.5	6824.7	7328.5	7857.6

这是一个一维数值插值问题，命令如下：

```
T=0:5:65;
X=2:5:57;
F=[3.2015,2.2560,879.5,1835.9,2968.8,4136.2,5237.9,6152.7,...
     6725.3,6848.3,6403.5,6824.7,7328.5,7857.6];
F1=interp1(T,F,X)                    %用线性方法插值
F1 =
  1.0e+003 *
     0.0028    0.3532    1.2621    2.2891    3.4358    4.5769
     5.6038    6.3817    6.7745    6.6704    6.5720    7.0262
F1=interp1(T,F,X,'nearest')          %用最近方法插值
F1 =
  1.0e+003 *
     0.0032    0.0023    0.8795    1.8359    2.9688    4.1362
     5.2379    6.1527    6.7253    6.8483    6.4035    6.8247
F1=interp1(T,F,X,'spline')           %用三次样条方法插值
F1 =
  1.0e+003 *
    -0.1702    0.3070    1.2560    2.2698    3.4396    4.5896
     5.6370    6.4229    6.8593    6.6535    6.4817    7.0441
F1=interp1(T,F,X,'cubic')            %用三次多项式方法插值
F1 =
  1.0e+003 *
    -0.1026    0.2861    1.2479    2.2747    3.4364    4.5906
     5.6337    6.4280    6.8341    6.6697    6.5057    7.0191
```

2. 二维数值插值

当采样变量只与一个参数相关时，可以选定一个检测区间，记录下采样值，并根据对参数的采样值来构造一个一维函数，从而应用该函数计算参数在插值点的值，这就是一维插值问题。自然，某采样变量依据两个参数变化时，其采样点就应该是一个由这两个参数组成的一个平面区域，插值函数也是一个二维函数。对依赖于两个参数的函数进行插值的问题称为二维插值问题。同样，MATLAB 中，提供了解决二维插值问题的函数。其调用格式为：

```
ZI=interp2(X,Y,Z,X1,Y1,'method')
```

其中 X、Y 是两个向量，分别描述两个参数的采样点，Z 是与参数采样点对应的采样变量的样本值，$X1$、$Y1$ 是两个向量或标量，描述欲插值的点。method 的取值与一维插值函数相同。

同样，$X1$、$Y1$ 的取值范围不能超出 X、Y 的给定范围，否则，会给出"NaN"错误。

例 5.31　设 $Z=x^2+y^2$，对 Z 函数在 $(0,1)\times(0,2)$ 区域内进行插值。

命令如下：

```
x=0:0.1:10;y=0:0.2:20;
[X,Y]=meshgrid(x,y);
Z=X.^2+Y.^2;
```

```
interp2(x,y,Z,0.5,0.5)                    %对函数在(0.5,0.5)点进行插值
ans =
    0.5100
interp2(x,y,Z,[0.5 0.6],0.4)              %对函数在(0.5,0.4)点和(0.6,0.4)点进行插值
ans =
    0.4100    0.5200
interp2(x,y,Z,[0.5 0.6],[0.4 0.5])        %对函数在(0.5,0.4)点和(0.6,0.5)点进行插值
ans =
    0.4100    0.6200
interp2(x,y,Z,[0.5 0.6]',[0.4 0.5])  %对函数在(0.5,0.4),(0.6,0.4),(0.5,0.5)和(0.6,0.5)点进行插值
ans =
    0.4100    0.5200
    0.5100    0.6200
```

如果想提高插值精度,可选择插值方法为'spline',对于本例,精度可以提高。输入命令:

```
interp2(x,y,Z,[0.5 0.6]',[0.4 0.5],'spline')
ans =
    0.4100    0.5200
    0.5000    0.6200
```

3. 三维数值插值

对三维函数插值的函数是 interp3,其使用方法和 interp2 相同。其调用格式为:

```
W1 = interp3(X,Y,Z,W,X1,Y1,Z1,'method')
```

函数返回三维插值结果。其中 ***X***、***Y***、***Z*** 是 3 个向量,分别描述 3 个参数的采样点,*W* 是与参数采样点对应的采样变量的样本值,*X*1、*Y*1、*Z*1 是 3 个向量或标量,描述欲插值的点。method 是插值方法,可选,其取值与一、二维插值函数相同。

注意:*X*1、*Y*1、*Z*1 的取值范围不能超出 ***X***、***Y***、***Z*** 的给定范围,否则,会给出"NaN"错误。

5.4.3 曲线拟合

1. 曲线拟合与最小二乘法

与数值插值类似,曲线拟合的目的也是用一个较简单的函数去逼近一个复杂的或未知的函数,所依据的条件都是在一个区间(或一个区域)上的有限个采样点的函数值。数值插值要求逼近函数在采样点与被逼近函数相等,在要求逼近函数是一个统一(而不是分段函数)的较简单的函数(如多项式)时,并非总能满足;曲线拟合放弃在插值点两者完全相等的要求,而只要求两者之差的平方和最小,这就是最小二乘法的基本思想。设 $f(x)$ 是被插值函数,$p(x)$ 是一个多项式,在采样点的取值如表 5.2 所示。

表 5.2 函数采样示意表

x	x_1	x_2	x_3	…	x_n
$f(x)$	f_1	f_2	f_3	…	f_n
$p(x)$	p_1	p_2	p_3	…	p_n

当 $\sum_{i=1}^{n}(f_i - p_i)^2$ 最小时，称 $p(x)$ 是 $f(x)$ 的一个最佳平方逼近（也称为最小二乘逼近）。数学上已经证明，上述最小二乘逼近问题的解总是确定的。有关的证明推导过程，有兴趣的读者请参考有关资料。

2. 曲线拟合的实现

曲线采用最小二乘法则进行拟合时，实际上是求一个系数向量，该系数向量是一个多项式的系数。MATLAB 中，提供了解决使用最小二乘法进行曲线拟合的函数。调用格式为：

```
[P,S] = polyfit(X,Y,m)
```

函数根据采样点 X 和采样点函数值 Y，产生一个 m 次多项式 P 及其在采样点的误差向量 S。其中 X、Y 是两个等长的向量，P 是一个长度为 $m+1$ 的向量。

例 5.32　用一个5次多项式在区间[0,2π]内逼近函数 sin(x)。

在给定区间上，均匀地选择 50 个采样点 X，并应用 sin(X)计算采样点函数值 Y。然后利用 5 次多项式逼近。命令如下：

```
X = linspace(0,2 * pi,50); Y = sin(X);
[P,S] = polyfit(X,Y,5)                    %得到 5 次多项式的系数和误差
P =
  -0.0056    0.0874    -0.3946    0.2685    0.8797    0.0102
S =
          R: [6x6 double]
         df: 44
      normr: 0.0337
plot(X,Y,'k*',X,polyval(P,X),'k-')
```

绘出 sin(X)和多项式 P(X)在给定区间的函数曲线（图 5.1 所示），其中 * 曲线是 sin(X)，实

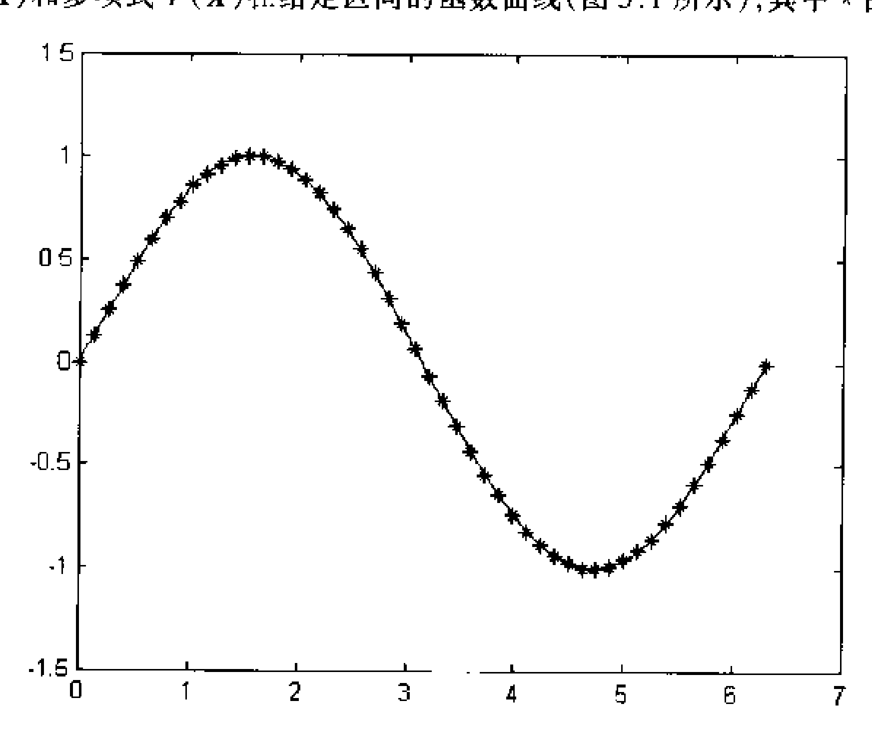


图 5.1　用 5 次多项式对正弦函数的拟合

线是 $P(x)$。函数 polyval(P,X)是一个 MATLAB 函数,返回多项式 $P(X)$的值,下一节将作介绍。

从图 5.1 可以看出,5 次多项式对正弦函数的拟合效果是非常好的。

5.4.4 多项式计算

多项式是一种最简单的函数,也是用来对其他复杂函数进行逼近的工具。这一节讨论 MATLAB 中多项式的各种运算函数。

一个 m 次多项式有 $m+1$ 个系数,在 MATLAB 中,用一个长度为 $m+1$ 的向量表示一个 m 次多项式。

1. 多项式的建立

(1) 已知多项式的全部系数建立多项式

当已知多项式的全部系数时,建立多项式和建立一个向量的方法完全相同,直接用赋值语句即可建立。注意变量最高次幂的系数作为向量的第一个分量,而常数项作为最后一个分量,不要弄反。

(2) 已知多项式的全部根建立多项式

已知一个多项式的全部根 **X** 求多项式系数的函数是 poly(X),该函数返回以 **X** 为全部根的一个多项式 **P**,当 **X** 是一个长度为 m 的向量时,**P** 是一个长度为 $m+1$ 的向量。

2. 多项式求根

求多项式 $p(x)$的根的函数是 roots(P),这里,P 是 p(x)的系数向量,该函数返回方程 $p(x)=0$ 的全部根(含重根,复根)。

3. 多项式求值

求多项式 $p(x)$在某点或某些点的函数值的函数是 polyval(P,x)。若 x 为一数值,则求多项式在该点的值;若 x 为向量或矩阵,则对向量或矩阵中的每个元素求其多项式的值。

例 5.33 已知

$$f(x)=3x^5+4x^3-5x^2-7.2x+5$$

(1) 计算 $f(x)=0$ 的全部根。

(2) 由方程 $f(x)=0$ 的根构造一个多项式 $g(x)$,并与 $f(x)$进行对比。

(3) 计算 $f(5)$、$f(7.8)$、$f(9.6)$、$f(12.3)$的值。

命令如下:

```
P=[3,0,4,-5,-7.2,5];
X=roots(P)                    %求方程 f(x)=0 的根
X =
   -0.3046 + 1.6217i
   -0.3046 - 1.6217i
    1.0190
   -1.0066
    0.5967
G=poly(X)                      %求多项式 g(x)
G =
    1.0000    -0.0000    1.3333    -1.6667    -2.4000    1.6667
```

这是多项式 $f(x)$ 除以首项系数 3 的结果，两者的零点相同。

```
X0 = [5,7.8,9.6,12.3];
f = polyval(P,X0)                %求多项式 f(x)在给定点的值
f =
   1.0e+005 *
      0.0972    0.8816    2.4763    8.5120
```

多项式求值还有一个函数是 polyvalm，其调用格式与 polyval 相同，但含义不同。polyvalm 函数要求 x 为方阵，它以方阵为自变量求多项式的值。设 A 为方阵，P 代表多项式 x^3-5x^2+8，那么 polyvalm(P,A)的含义是：

A * A * A - 5 * A * A + 8 * eye(size(A))

而 polyval(P,A)的含义是：

A . * A . * A - 5 * A . * A + 8 * ones(size(A))

读者不妨上机验证一下。

4. 多项式的四则运算

(1) 多项式的加减法

MATLAB 中，未提供专门多项式加减法函数，事实上，多项式的加减法就是其所对应的向量的加减法。当两个多项式具有相同的次数时，对两个系数向量直接进行加减法；当两个多项式的次数不同时，要在一个较低次幂的多项式系数向量前补 0，使两个系数向量等长。

(2) 多项式的乘法

函数 conv(P1,P2)用于求多项式 $P1$ 和 $P2$ 的乘积。这里，$P1$、$P2$ 是两个多项式系数向量。

(3) 多项式的除法

函数[Q,r] = deconv(P1,P2)用于对多项式 $P1$ 和 $P2$ 做除法运算。其中 Q 返回多项式 $P1$ 除以 $P2$ 的商式，r 返回 $P1$ 除以 $P2$ 的余式。这里，Q 和 r 仍是多项式系数向量。

deconv 是 conv 的逆函数，即有 P1 = conv(P2,Q) + r。

例 5.34　设

$f(x)=3x^5-5x^4+2x^3-7x^2+5x+6$

$g(x)=3x^2+5x-3$

(1) 求 $f(x)+g(x)$ 和 $f(x)-g(x)$。

(2) 求 $f(x)g(x)$ 和 $f(x)/g(x)$。

在 MATLAB 命令窗口，输入命令：

```
f = [3,-5,2,-7,5,6];g = [3,5,-3];g1 = [0,0,0,g];
f + g1                           %求 f(x) + g(x)
ans =
    3    -5    2    -4    10    3
f - g1                           %求 f(x) - g(x)
ans =
    3    -5    2    -10    0    9
conv(f,g)                        %求 f(x) * g(x)
ans =
```

```
     9    0    -28    4    -26    64    15    -18
[Q,r] = deconv(f,g)                %求 f(x)/g(x),商式送 Q,余式送 r。
Q =
   1.0000    -3.3333    7.2222    -17.7037
r =
   0    0         0          0    115.1852    -47.1111
```

5. 多项式的导函数

对多项式求导数的函数是:

p = polyder(P)　求多项式 P 的导函数

p = polyder(P,Q)　求 P * Q 的导函数

[p,q] = polyder(P,Q)　求 P/Q 的导函数,导函数的分子存入 p,分母存入 q。

上述函数中,参数 P、Q 是多项式的向量表示,结果 p、q 也是多项式的向量表示。

例 5.35 求有理分式

$$f(x)=\frac{3x^5+5x^4-8x^2+x-5}{10x^{10}+5x^9+6x^6+7x^3-x^2-100}$$

的导数。

命令如下:

```
P = [3,5,0,-8,1,-5];
Q = [10,5,0,0,6,0,0,7,-1,0,-100];
[p,q] = polyder(P,Q)
p =
          -150    -360    -125    640     172     400
           225     234      -4    170   -1444   -2014
           106    1590    -100
q =
           100     100      25      0     120      60
             0     140      86    -10   -2000    -916
           -12       0   -1151    -14       1   -1400
           200       0   10000
```

导数的分子是一个 14 次多项式,其 14 次幂的系数是 -150,常数项是 -100;分母是一个 20 次多项式,其 20 次幂的系数是 100,常数项是 10000。

5.4.5 函数的最大值与最小值

求函数在指定区间的最大值与最小值是一种常见的运算,MATLAB 中用于求最小值的函数是:

fmin(f,a,b)　求单变量函数 $f(x)$在区间(a,b)上的最小值点。

fmins(F,X0)　求多变量函数 $F(x)$在估计值 $X0$ 附近的最小值点。

MATLAB 没有专门提供求函数最大值点的函数,但只要注意到 $-f(x)$在区间(a,b)上的最小值点就是 $f(x)$在(a,b)的最大值点,所以 fmin(-f,a,b)返回函数 $f(x)$在区间(a,b)上的最大值。

例 5.36 求函数

$$f(x)=x-\frac{1}{x}+5$$

在区间(-10,1)和(1,10)上的最小值点。

首先建立函数文件 fx.m:

```
function f = f(x)
f = x - 1/x + 5;
return
```

上述函数文件也可用一个语句函数代替:

```
ff = inline('x - 1/x + 5')
```

再在 MATLAB 命令窗口,输入命令:

```
fmin('fx', -10, -1)                 %求函数在区间(-10, -1)内的最小值点
ans =
    -9.9999
fmin(ff,1,10)                       %求函数在区间(1,10)内的最小值点。注意函数名 ff 不用加'
ans =
    1.0001
```

例 5.37 设

$$f(x,y,z)=x+\frac{y^2}{4x}+\frac{z^2}{y}+\frac{2}{z}$$

求函数 f 在(0.5,0.5,0.5)附近的最小值。

建立函数文件 fxyz.m:

```
function f = f(u)
x = u(1);y = u(2);z = u(3);
f = x + y.^2./x/4 + z.^2./y + 2./z;
return
```

在 MALAB 命令窗口,输入命令:

```
U = fmins('fxyz',[0.5,0.5,0.5])          %求函数的最小值点
U =
  0.5000      1.0000      1.0000
fxyz(U)                                  %求函数的最小值
ans =
    4.0000
```

5.5 傅立叶分析

离散傅立叶变换在通信、数据测试与分析及过程控制中有广泛的应用。本节先简要阐述离散傅立叶变换的基本概念和变换公式,然后讨论 MATLAB 中离散傅立叶变换的实现。

1. 离散傅立叶变换的基本概念

设向量 $X=(x_1,x_2,\cdots,x_N)$是某个参数的一组样本值,称向量 $Y=(y_1,y_2,\cdots,y_N)$为 X 的一

个离散傅立叶变换，其中

$$y_k = \sum_{m=1}^{N} x_m e^{j2\pi(m-1)k/N}$$

称上式为离散傅立叶变换公式，而下式称为离散傅立叶逆变换公式。

$$x_k = \frac{1}{N}\sum_{m=1}^{N} y_m e^{-j2\pi(m-1)k/N}$$

2. 离散傅立叶变换的实现

MATLAB 中，提供了对向量(或直接对矩阵的行或列)进行离散傅立叶变换的函数，其调用格式是：

```
Y = fft(X,n,dim)
```

(1) 当 X 是一个向量时，返回对 X 的离散傅立叶变换。n 可以省略，其缺省值为 X 的长度 m = length(X)，$n < m$ 时，只截取前 n 个元素进行变换；$n \geq m$ 时，在 X 后面补上 $n - m$ 个 0 元素。X 是一个向量时，参数 *dim* 不选。

(2) 当 X 是一个矩阵时，返回一个矩阵并送 Y，其列(行)是对 X 的列(行)的离散傅立叶变换。参数 n 与(1)相同，$dim = 1$ 时，对 X 进行列变换；$dim = 2$ 时，对 X 进行行变换。*dim* 的缺省值是 1。

例 5.38　求 $X = (1,0,-3,5,2)$的离散傅立叶逆变换。

在 MATLAB 命令窗口，输入命令：

```
X = [1,0,-3,5,2];
Y = fft(X)              % 对 X 进行变换
Y =
    5.0000    -0.0000 + 6.6044i    0.0000 - 6.4329i    -0.0000 + 6.4329i
    0.0000 - 6.6044i
```

3. 离散傅立叶变换的逆变换

MATLAB 中，对向量(或直接对矩阵的行或列)进行离散傅立叶逆变换的函数的调用方法是：

```
Y = ifft(X,n,dim)
```

函数对 X 进行离散傅立叶逆变换。其中 X、n、*dim* 的意义及用法和离散傅立叶变换函数 fft 完全相同。

例 5.39　对矩阵

$$A = \begin{bmatrix} 3 & 2 & 1 & 1 \\ -5 & 1 & 0 & 1 \\ 3 & 2 & 1 & 5 \end{bmatrix}$$

的列向量、行向量分别进行离散傅立叶变换并对变换结果进行逆变换。

命令如下：

```
A = [3,2,1,1;-5,1,0,1;3,2,1,5];
fftA = fft(A)              % 求 A 的列向量的傅立叶变换
fftA =
     1.0000             5.0000             2.0000             7.0000
     4.0000 + 6.9282i   0.5000 + 0.8660i   0.5000 + 0.8660i   -2.0000 + 3.4641i
     4.0000 - 6.9282i   0.5000 - 0.8660i   0.5000 - 0.8660i   -2.0000 - 3.4641i
```

```
fftA2 = fft(A,4,2)          %求 A 的行向量的傅立叶变换
fftA2 =
      7.0000    2.0000 - 1.0000i   1.0000    2.0000 + 1.0000i
     -3.0000   -5.0000            -7.0000   -5.0000
     11.0000    2.0000 + 3.0000i  -3.0000    2.0000 - 3.0000i
ifft(fftA)                  %对矩阵 fftA 的列向量进行傅立叶逆变换,结果应等于 A
ans =
   3.0000 - 0.0000i  2.0000 - 0.0000i  1.0000 - 0.0000i  1.0000 - 0.0000i
  -5.0000            1.0000 - 0.0000i  0 - 0.0000i       1.0000 - 0.0000i
   3.0000 - 0.0000i  2.0000 + 0.0000i  1.0000 + 0.0000i  5.0000 + 0.0000i
ifft(fftA2,4,2)             %对矩阵 fftA2 的行向量进行傅立叶逆变换,其结果应等于 A
ans =
    3.0000   2.0000 + 0.0000i  1.0000   1.0000 - 0.0000i
   -5.0000   1.0000                 0   1.0000
    3.0000   2.0000 - 0.0000i  1.0000   5.0000 + 0.0000i
```

5.6　数值微积分

5.6.1　数值微分

一般来说,函数的导数依然是一个函数。数值方法的基本思想是:不求问题的解析表达结果,而只求问题对应于某些点的解,称这样离散的解为数值解。设函数 $f(x)$ 的导函数 $f'(x)=g(x)$,高等数学关心的是 $g(x)$ 的形式和性质,而数值分析关心的问题是怎样计算 $g(x)$ 在一串离散点 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 的近似值 $\boldsymbol{G}=(g_1,g_2,\cdots,g_n)$ 以及所计算的近似值有多大误差。误差可以两个向量差的范数描述:

$$\Delta=\|(g(x_1),g(x_2),\cdots,g(x_n))-(g_1,g_2,\cdots,g_n)\|$$

当然,进行误差分析必须对 $g(x)$ 的性质有所了解。

1. 数值差分与差商

任意函数 $f(x)$ 在 x 点的导数是通过极限定义的

$$\begin{cases}f'(x)=\lim\limits_{h\to0}\dfrac{f(x+h)-f(x)}{h}\\ f'(x)=\lim\limits_{h\to0}\dfrac{f(x)-f(x-h)}{h}\\ f'(x)=\lim\limits_{h\to0}\dfrac{f(x+h/2)-f(x-h/2)}{h}\end{cases}$$

上述式子中,均假设 $h>0$,如果去掉上述等式右端的 $h\to0$ 的极限过程,并引进记号

$$\begin{cases}\Delta f(x)=f(x+h)-f(x)\\ \nabla f(x)=f(x)-f(x-h)\\ \delta f(x)=f(x+h/2)-f(x-h/2)\end{cases}$$

称 $\Delta f(x)$、$\nabla f(x)$及 $\delta f(x)$分别为函数在 x 点处以 $h(h>0)$为步长的向前差分、向后差分和中心差分。当步长 h 充分小时，有理由相信：

$$\begin{cases} f'(x) \approx \dfrac{\Delta f(x)}{h} \\ f'(x) \approx \dfrac{\nabla f(x)}{h} \\ f'(x) \approx \dfrac{\delta f(x)}{h} \end{cases}$$

和差分一样，称 $\Delta f(x)/h$、$\nabla f(x)/h$ 及 $\delta f(x)/h$ 分别为函数在 x 点处以 $h(h>0)$为步长的向前差商、向后差商和中心差商。当步长 $h(h>0)$充分小时，函数 f 在点 x 的微分接近于函数在该点的任意种差分，而 f 在点 x 的导数接近于函数在该点的任意种差商。

2. 数值微分的实现

有两种方式计算任意函数 $f(x)$在给定点 x 的数值导数，第 1 种方式是用多项式或样条函数 $g(x)$对 $f(x)$进行逼近(插值或拟合)，然后用逼近函数 $g(x)$在点 x 处的导数作为$f(x)$在点 x 处的导数。第 2 种方式是用 $f(x)$在点 x 处的某种差商作为其导数。MATLAB 中，没有直接提供求数值导数的函数，只有计算向前差分的函数。

DX = diff(X)　计算向量 $\boldsymbol{X}$ 的向前差分，DX(i) = X(i+1) - X(i)，0 < i < n。

DX = diff(X,n)　计算 $\boldsymbol{X}$ 的 n 阶向前差分，diff(X,2) = diff(diff(X))。

DX = diff(A,n,dim)　计算矩阵 $\boldsymbol{A}$ 的 n 阶差分，$dim=1$ 时(缺省状态)，按列计算差分；$dim=2$，按行计算差分。

例 5.40　求向量 sin($\boldsymbol{X}$)的 1 ~ 3 阶差分。设 $\boldsymbol{X}$ 由$[0,2\pi]$间均匀分布的 10 个点组成。

命令如下：

```
X = linspace(0,2 * pi,10);
Y = sin(X);
DY = diff(Y);                  %计算 Y 的一阶差分
D2Y = diff(Y,2);               %计算 Y 的二阶差分,也可用命令 diff(DY)计算
D3Y = diff(Y,3);               %计算 Y 的三阶差分,也可用 diff(D2Y)或 diff(DY,2)
```

输出结果分别是：

```
X =
        0   0.6981   1.3963   2.0944   2.7925   3.4907   4.1888
   4.8869   5.5851   6.2832
Y =
         0    0.6428    0.9848    0.8660   0.3420   -0.3420   -0.8660
   -0.9848   -0.6428   -0.0000
DY =
    0.6428   0.3420   -0.1188   -0.5240   -0.6840   -0.5240   -0.1188
    0.3420   0.6428
D2Y =
    -0.3008   -0.4608   -0.4052   -0.1600   0.1600   0.4052   0.4608
     0.3008
```

```
D3Y =
   -0.1600    0.0556    0.2452    0.3201    0.2452    0.0556   -0.1600
```

例 5.41　用不同的方法求函数 $f(x)$ 的数值导数，并在同一个坐标系中做出 $f'(x)$ 的图像。设

$$f(x)=\sqrt{x^3+2x^2-x+12}+\sqrt[6]{x+5}+5x+2$$

为进行对比，求出 $f'(x)$

$$f'(x)=\frac{3x^2+4x-1}{2\sqrt{x^3+2x^2-x+12}}+\frac{1}{6\sqrt[6]{(x+5)^5}}+5$$

为确定计算数值导数的点，假设在[-3,3]区间以 0.01 为步长求数值导数，下面用 3 种方法求 $f(x)$ 在这些点的导数。首先用一个 5 次多项式 $p(x)$ 拟合函数 $f(x)$，并对 $p(x)$ 求一般意义下的导数，得 $dp(x)$，求出 $dp(x)$ 在假设点的值；第 2 种方法直接求 $f(x)$ 在假设点的数值导数；第 3 种方法直接求 $f'(x)$ 在假设点的导数，最后用一个坐标图显示这 3 条曲线。程序如下：

```
f = inline('sqrt(x.^3 + 2 * x.^2 - x + 12) + (x + 5).^(1/6) + 5 * x + 2');
g = inline('(3 * x.^2 + 4 * x - 1)./sqrt(x.^3 + 2 * x.^2 - x + 12)/2 + 1/6./(x + 5).^(5/6) + 5');
x = -3:0.01:3;
p = polyfit(x,f(x),5);                  %用 5 次多项式 p 拟合 f(x)
dp = polyder(p);                        %对拟合多项式 p 求导数 dp
dpx = polyval(dp,x);                    %求 dp 在假设点的函数值
dx = diff(f([x,3.01]))/0.01;            %直接对 f(x)求数值导数
gx = g(x);                              %求函数 f 的导函数 g 在假设点的导数
plot(x,dpx,x,dx,'g.',x,gx,'r-');        %作图
```

程序运行后得到图 5.2 所示的图形。

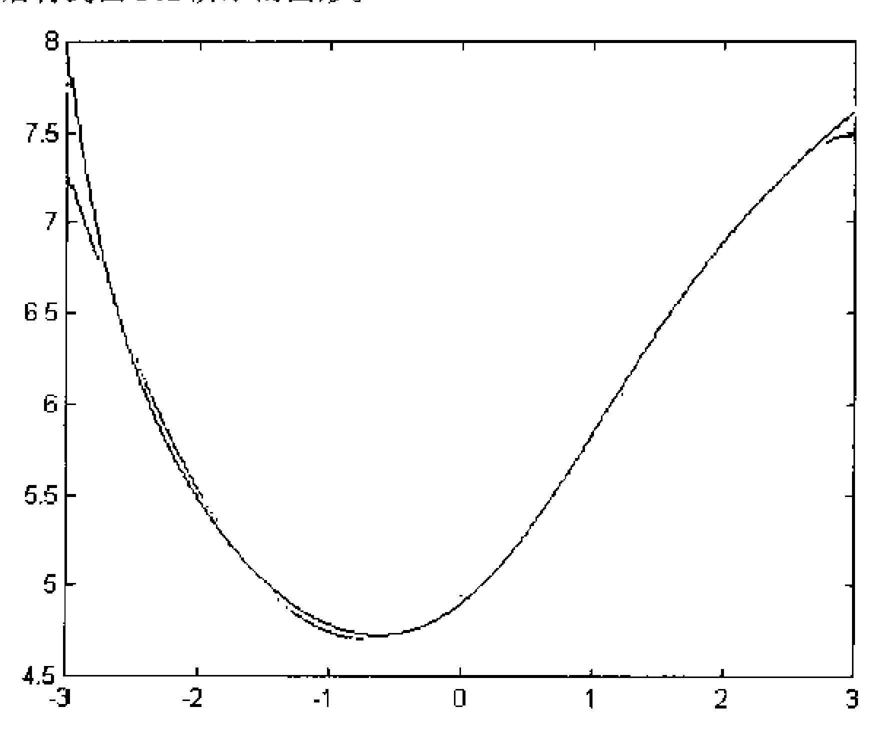


图 5.2　用不同方法求得的数值导数

5.6.2 数值积分

1. 数值积分的基本原理

数值积分研究定积分的数值求解方法。设

$$I1 = \int_a^b f(x)\mathrm{d}x, \quad I2 = \int_a^b p(x)\mathrm{d}x$$

从高等数学中知道,当$|f(x) - p(x)| < \varepsilon$时,$|I1 - I2| < \varepsilon(b-a)$。这说明,当$\varepsilon$充分小时,可用$I2$近似地代替$I1$。所以,求任意函数$f(x)$在$[a,b]$上的定积分时,当难以使用解析的方法求出$f(x)$的原函数时,则可以寻找一个在$[a,b]$上与$f(x)$逼近,但形式上却简单且易于求积的函数$p(x)$,用$p(x)$在$[a,b]$上的积分值近似地代替$f(x)$在$[a,b]$上的积分值。一般选择被积函数的插值多项式充当这样的替代函数。选择的插值多项式的次数不同,就形成了不同的数值积分公式。选择一次多项式时,称为梯形公式,选择二次多项式时,称为辛普生(Simpson)公式。

如果把积分区间$[a,b]$分划为n个等长的子区间

$$[a,b] = [a,a_1] \cup [a_1,a_2] \cup \cdots \cup [a_{n-1},b]$$

则在每个子区间$[a_i,a_{i+1}]$上用$f(x)$的插值多项式$p(x)$代替$f(x)$,其逼近效果一般将会比在整个区间上使用一个统一的插值多项式时更好。这样就形成了数值积分复合公式。对被积函数$f(x)$采用一、二次多项式插值,然后对插值多项式求积分,就得到了几个常见的数值积分公式

$$S_1 = \frac{b-a}{2}[f(a) + f(b)]$$

$$S_2 = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

$$S_3 = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(a+ih)\right]$$

$$S_4 = \frac{h}{6}\sum_{i=0}^{n-1}\left[f(a+ih) + 4f\left(a + \frac{i+1}{2}h\right) + f(a+(i+1)h)\right]$$

S_1、S_2是基本梯形和基本辛普生求积公式,S_3、S_4是复合梯形和复合辛普生求积公式。计算数学中已经证明,对一般工程问题,复合辛普生积分公式具有足够的精度。

2. 数值积分的实现

被积函数一般是用一个解析式给出,但也有很多情况下用一个表格形式给出。MATLAB 中,对这两种给定被积函数的方法,提供了不同的数值积分函数。

(1) 被积函数是一个解析式

函数 quad(f,a,b,tol,trace)用于求被积函数$f(x)$在$[a,b]$上的定积分,*tol*是计算精度,缺省值是 0.001。*trace* 非 0 时,画出积分图形。注意,调用 quad 函数时,先要建立一个描述被积函数$f(x)$的函数文件或语句函数。当被积函数 f 含有一个以上的变量时,quad 函数的调用格式为:

```
quad(f,a,b,tol,trace,g1,g2)
```

其中f,a,b,*tol*,*trace* 等参数的含义同前。$g1$,$g2$代人f的非积分变量中,即将$f = f(x,y,z)$在$[a,b]$上的积分确定为$f(g1,g2,z)$在$[a,b]$上的积分。当省略 *tol*,trace 时,要补上[],具体用法见例 5.44。

数值积分函数还有一种形式 quad8,其用法与 quad 完全相同。一般说来,quad8 比 quad 精度

要高。

例 5.42　用两种不同的方法求

$$I = \int_0^1 e^{-x^2} dx$$

先建立一个函数文件 ex.m：

```
function ex = ex(x)
ex = exp( - x.^2);                    %注意应用点运算
return
```

然后，在 MATLAB 命令窗口，输入命令：

```
quad(' ex ',0,1,1e - 6)               %注意函数名应加字符引号
ans =
     0.7468
quad8(' ex ',0,1,1e - 6)              %用另一函数求积分
ans =
     0.7468
```

也可不建立关于被积函数的函数文件，而使用语句函数(内联函数)求解，命令如下：

```
g = inline(' exp( - x.^2)');           %定义一个语句函数 g(x) = exp( - x^2)
quad8(g,0,1)                           %注意函数名不加'号
ans =
     0.7468
```

读者可用解析方法计算一下本例，并与本例所显示的结果做一比较。

(2) 被积函数由一个表格定义

在科学实验和工程应用中，函数关系往往是不知道的，只有实验测定的一组样本点和样本值，这时，人们就无法使用 quad 函数计算其定积分。MATLAB 中，对由表格形式定义的函数关系求定积分的问题来用 trapz(X,Y)函数。其中向量 $\boldsymbol{X}$、$\boldsymbol{Y}$ 定义函数关系 $\boldsymbol{Y}=f(\boldsymbol{X})$。$\boldsymbol{X}$、$\boldsymbol{Y}$ 是两个等长的向量：$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$，$\boldsymbol{Y}=(y_1,y_2,\cdots,y_n)$，并且 $x_1<x_2<\cdots<x_n$，积分区间是$[x_1,x_n]$。

例 5.43　用 trapz 函数计算

$$I = \int_0^1 e^{-x^2} dx$$

在 MATLAB 命令窗口，输入命令：

```
X = 0:0.01:1;Y = exp( - X.^2);
trapz(X,Y)
ans =
      0.7468
```

(3) 二重积分

例 5.44　计算

$$I = \int_0^1 \int_0^1 e^{-x^2-y^2} dx dy$$

的二重积分。

MATLAB 中虽然直接提供计算二重积分的函数，但这里首先用累次积分的方法完成本例。

本例积分区域是一个矩形,可以很容易将其转化为二次积分进行处理。具体方法是:首先在 x 方向对区间[0,1]进行一个分划,得到一个向量,用 x 的每一个分量代入积分公式,此时所给积分化为一个关于 y 的定积分,用函数 quad 就可求出该积分,然后对这一列积分值使用函数 trapz 沿 x 方向进行第二次积分。

建立一个函数文件 fixy.m:

```
function f = f(x,y)
f = exp( - x.^2 - y.^2);
return
```

建立一个命令文件 ftxy1.m:

```
for i = 1:20
   int2(i) = quad(' fixy ',0,1,[],[],x(i));   %在二维函数 fixy 中以 x = x(i)代入并对 y 积分。
end
```

在 MATLAB 命令窗口,输入命令:

```
x = linspace(0,1,20);
ftxy1
trapz(x,int2)
ans =
   0.5576
```

读者会觉得上述过程非常烦琐,实际上,MATLAB 提供了计算二重积分的函数:

```
dblquad(f,a,b,c,d,tol,trace)
```

该函数求 $f(x,y)$ 在 $[a,b]\times[c,d]$ 区域上的二重积分。参数 *tol*, *trace* 的用法与函数 quad 完全相同。

如果直接使用这里介绍的二重积分函数 dblquad 来求解本例就非常简单,命令如下:

```
g = inline(' exp( - x.^2 - y.^2)');
dblquad(g,0,1,0,1)                    %直接调用二重积分函数求解
ans =
     0.5577
```

5.7 常微分方程的数值求解

5.7.1 引言

这里以一阶常微分方程为例叙述常微分方程的基本概念。一阶常微分方程的初值问题的一般形式是

$$y' = f(x,y)$$
$$y(x_0) = y_0$$

多数情况下,一阶常微分初值问题的解是存在且惟一的,但存在解并不意味着可以用有限形式表示其解。事实上,在大多数情况下,一阶常微分初值问题的解是不能用有限形式表示的。我

们知道,常微分方程的解是一个函数,既然不能找到解的解析表达式,能否像数值微分一样即不求解函数的一般表达式,而只求解函数在指定点的函数值,这就导致了对微分方程数值解的研究。本节对微分方程数值解的基本思想做一个简单的介绍,并在此基础上讨论 MATLAB 中的常微分方程初值问题的数值解的求解方法。

5.7.2 龙格—库塔法简介

常微分方程初值问题的数值解的方法很多,有欧拉(Euler)法、线性多步法、预估校正法、龙格—库塔法(Runge - Kutta)等,以龙格—库塔法使用最多,这里只对该方法进行介绍。MATLAB 中,也采用龙格—库塔方法实现对常微分方程初值问题的数值解求解。

为了叙述方便,这里重写一次一阶常微分方程的初值问题的一般形式

$$y' = f(x, y)$$
$$y(x_0) = y_0$$

在求解未知函数 y 时,y 在 x_0 点的值 $y(x_0) = y_0$ 是已知的,并且根据高等数学中的中值定理,应有

$$y(x_0 + h) = y_1 \approx y_0 + hf(x_0, y_0), h > 0, \text{称为步长}$$
$$y(x_0 + 2h) = y_2 \approx y_1 + hf(x_1, y_1)$$

一般地,在任意点 $x_i = x_0 + ih$,有

$$y(x_0 + ih) = y_i \approx y_{i-1} + hf(x_{i-1}, y_{i-1}), i = 1, 2, \cdots, n$$

当(x_0, y_0)确定后,根据上述递推式能计算出未知函数 y 在点 $x_i = x_0 + ih, i = 0, 1, \cdots, n$ 的一列数值解

$$y_i = y_0, y_1, y_2, \cdots, y_n, i = 0, 1, \cdots, n$$

当然,递推过程中有一个误差累计的问题。在实际计算过程中,使用的递推公式一般进行过改造,著名的龙格—库塔公式是

$$y(x_0 + ih) = y_i = y_{i-1} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

其中

$$k_1 = f(x_{i-1}, y_{i-1})$$
$$k_2 = f(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}hk_1)$$
$$k_3 = f(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}hk_2)$$
$$k_4 = f(x_{i-1} + h, y_{i-1} + hk_3)$$

5.7.3 龙格—库塔法的实现

基于龙格　库塔法,MATLAB 提供了求常微分方程数值解的函数,一般调用格式为:

```
[X,Y] = ode23(f,[x0,xn],y0)
[X,Y] = ode45(f,[x0,xn],y0)
```

其中 $\boldsymbol{X}$、$\boldsymbol{Y}$ 是两个向量,$\boldsymbol{X}$ 对应自变量 x 在求解区间$[x0, xn]$的一组采样点,其采样密度是自适应的,无需指定;$\boldsymbol{Y}$ 是与 $\boldsymbol{X}$ 对应的一组解,f 是一个函数,$[x0, xn]$代表自变量的求解区间,$y0 =$

$y(x0)$,由方程的初值给定。函数在求解区间$[x0,xn]$内,自动设立采样点向量 X,并求出解函数 y 在采样点 X 处的样本值。

这两个函数分别采用了二阶、三阶龙格—库塔法和四阶、五阶龙格—库塔法,并采用自适应变步长的求解方法,即当解的变化较慢时采用较大的步长,从而使得计算速度很快。当解的变化较快时步长会自动地变小,从而使得计算精度很高。

例 5.45 求微分方程

$$\begin{cases}\dfrac{dy}{dx}=\dfrac{-2y}{x}+4x\\ y(1)=2\end{cases}$$

初值问题在[1,3]区间内的数值解,并将结果与解析解($y=x^2+\frac{1}{x^2}$)进行比较。

这里 $f(x,y)=-2y/x+4x$,先建立一个该函数的 M 文件 fxy1.m:

```
function f = f(x,y)
f= -2.*y./x+4*x          %注意使用点运算符
return
```

再输入命令:

```
[X,Y] = ode45('fxy1',[1,3],2);
X'                          %显示自变量的一组采样点
ans =
    1.0000  1.0500  1.1000  1.1500  1.2000  1.2500  1.3000
    1.3500  1.4000  1.4500  1.5000  1.5500  1.6000  1.6500
    1.7000  1.7500  1.8000  1.8500  1.9000  1.9500  2.0000
    2.0500  2.1000  2.1500  2.2000  2.2500  2.3000  2.3500
    2.4000  2.4500  2.5000  2.5500  2.6000  2.6500  2.7000
    2.7500  2.8000  2.8500  2.9000  2.9500  3.0000
Y'                    %显示求解函数与采样点对应的一组数值解,结果是:
ans =
    2.0000  2.0095  2.0364  2.0786  2.1344  2.2025  2.2817
    2.3712  2.4702  2.5781  2.6944  2.8187  2.9506  3.0898
    3.2360  3.3890  3.5486  3.7147  3.8870  4.0655  4.2500
    4.4405  4.6368  4.8388  5.0466  5.2600  5.4790  5.7036
    5.9336  6.1691  6.4100  6.6563  6.9079  7.1649  7.4272
    7.6947  7.9676  8.2456  8.5289  8.8174  9.1111
(X.^2+1./X.^2)'       %显示求解函数与采样点对应的一组解析解,结果是:
ans =
    2.0000  2.0095  2.0364  2.0786  2.1344  2.2025  2.2817
    2.3712  2.4702  2.5781  2.6944  2.8187  2.9506  3.0898
    3.2360  3.3890  3.5486  3.7147  3.8870  4.0655  4.2500
    4.4405  4.6368  4.8388  5.0466  5.2600  5.4790  5.7036
    5.9336  6.1691  6.4100  6.6563  6.9079  7.1649  7.4272
    7.6947  7.9676  8.2456  8.5289  8.8174  9.1111
```

从对两组结果的比较中可以看出,ode45 函数具有很高的精度。

例 5.46　求解

$$\begin{cases}\dfrac{d^2y}{dx^2}+x\dfrac{dy}{dx}-x^2+5=0\\ y(0)=5\\ y'(0)=6\end{cases}$$

初值问题在区间[0,2]中的解。

这是一个二阶常微分方程,ode45 函数只能求解一阶常微分方程和方程组,故应先将所给问题转化为一个等价的一阶微分方程组。

令 $u=y,\ v=y'$ 得

$$\begin{cases}\dfrac{dv}{dx}=-xv+x^2-5\\ \dfrac{du}{dx}=v\\ u(0)=5\\ v(0)=6\end{cases}$$

建立一个函数文件 fxy2.m:

```
function f = f(x,y)
f(2) = -x.*y(2) + x.^2 - 5;
f(1) = y(2);
f = f';
return
```

在 MATLAB 命令窗口,输入命令:

```
[X,Y] = ode45('fxy2',[0,2],[5,6]);
[X,Y]
ans =
        0   5.0000   6.0000
   0.0419   5.2467   5.7856
   0.0837   5.4843   5.5615
   0.1256   5.7123   5.3288
   0.1675   5.9304   5.0885
   0.2175   6.1774   4.7929
   0.2675   6.4095   4.4896
   0.3175   6.6263   4.1806
   0.3675   6.8275   3.8676
   0.4175   7.0130   3.5526
   0.4675   7.1828   3.2373
```

结果第一列为 x 的采样点,第二列和第三列分别为 y 和 y' 与 x 对应点的值(只列出部分结果)。

5.8 非线性方程的数值求解

在 5.3 节中，已经比较详细地讨论了线性代数方程组的解法。在实际应用过程中，还有大量的问题要归结于一个非线性方程或方程组的求解。非线性方程的求解过程比线性方程要困难很多，一般没有程序性的求解步骤，需要解题者的知识与技巧。本节讨论在 MATLAB 中怎样求解一般的非线性方程和方程组。在介绍 MATLAB 函数之前，为了使读者更好地明了函数参数的意义，先简要介绍所需的数学背景知识。

5.8.1 数学迭代法简介

单变量方程的一般形式是

$$f(x)=0$$

n 元方程组的一般形式是

$$\begin{cases} f_1(x_1,x_2,\cdots,x_n)=0 \\ f_2(x_1,x_2,\cdots,x_n)=0 \\ \vdots \\ f_n(x_1,x_2,\cdots,x_n)=0 \end{cases}$$

记

$$F(X)=\begin{bmatrix} f_1(X) \\ f_2(X) \\ \vdots \\ f_n(X) \end{bmatrix} \quad X=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad O=\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

称函数 $F(X)$为向量值函数。引进了上述向量记号后，一般 n 元方程组可以简记为

$$F(X)=O$$

向量值函数 $F(X)$中有一个分量函数 $f_i(X)$为非线性函数时，称方程组为非线性方程组。非线性方程组一般使用迭代法方法求解。迭代法方法的基本思想是，首先构造一个与方程组等价的方程组

$$X=G(X)$$

再选取一个初值向量 X_0，令 $X_1=G(X_0)$，$X_2=G(X_1)$，…

一般的有

$$X_{i+1}=G(X_i)$$

该式称为求解方程组的迭代式。由迭代式，可以从一个初值 X_0 出发，求得一列向量：X_0，X_1，X_2，…，X_n。对迭代式两边取极限，当极限

$$\lim_{n\to\infty} X_{n+1}=\lim_{n\to\infty} G(X_n)=X^*$$

存在时，显然，其极限值 X^* 就是方程组 $F(X)=0$ 的解。用这种方法求解方程的过程称为迭代法 。用迭代法求解方程组的关键问题是：如何构造一个函数 $G(X)$，使得：

(1) 迭代式是适当的,由其产生的一列 X_n 都在 $G(X)$的定义域内。

(2) 由其定义的向量列 X_n 收敛,即上述极限值存在。

(3) X_n 有较快的收敛速度。

上述过程对于一般人员是比较困难的。幸运的是计算数学工作者已经研究出了许多构造迭代函数 $G(X)$的方法,其中,牛顿(Newton)迭代法在解题过程中使用很普遍。牛顿迭代式是

$$X_{n+1} = X_n - [DF(X_n)]^{-1}F(X_n)$$

其中,$DF(X)$称为 $F(X)$的雅可比(Jacobi)矩阵

$$DF(X) = \begin{bmatrix} \frac{\partial f_1(X)}{\partial x_1} & \frac{\partial f_1(X)}{\partial x_2} & \cdots & \frac{\partial f_1(X)}{\partial x_n} \\ \frac{\partial f_2(X)}{\partial x_1} & \frac{\partial f_2(X)}{\partial x_2} & \cdots & \frac{\partial f_2(X)}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n(X)}{\partial x_1} & \frac{\partial f_n(X)}{\partial x_2} & \cdots & \frac{\partial f_n(X)}{\partial x_n} \end{bmatrix}$$

5.8.2　单变量非线性方程求解

在迭代式 $X_{i+1} = G(X_i)$中,设 $n = 1$,就得到了求解单变量方程 $f(x) = 0$ 的牛顿迭代式

$$X_{n+1} = X_n - \frac{f(x_n)}{f'(x_n)}$$

和求解微分方程一样,在 MATLAB 中,用户一般并不需要按上述迭代过程求解一个单变量方程,MATLAB 中,提供了求解单变量方程的函数 fzero(f,x0,tol),该函数采用迭代法计算函数 $f(x)$的一个零点,迭代初值为 x_0,当两次迭代结果小于 *tol* 时停止迭代过程。*tol* 的缺省值是 eps。

注意,在调用函数 fzero 之前,要使用 M 文件建立自己要计算的函数 $f(x)$,只有定义了函数 $f(x)$的 M 文件后,才能在 fzero 函数的参数中使用自定义函数名。

例 5.47　求 $f(x) = x - \frac{1}{x} + 5$ 在 $x_0 = -5$ 和 $x_0 = 1$ 作为迭代初值时的零点。

先编制一个函数文件 fz.m:

```
function f = f(x)
f = x - 1/x + 5;
```

然后,在 MATLAB 命令窗口,输入命令:

```
fzero('fz', -5)                    % 以 -5 作为迭代初值
Zero found in the interval: [ -4.8, -5.2].
ans =
    -5.1926
fzero('fz',1)
Zero found in the interval: [0.094903, 1.64].
ans =
     0.1926
```

5.8.3 非线性方程组求解

MATLAB 基本程序中未提供求解非线性方程组 $F(X)=0$ 的函数，但在其优化工具箱(Optimization Toolbox)中，为用户提供函数 fsolve，其调用格式为：

```
X = fsolve(F,X0)
```

其中，X、F 的意义如前述，$X0$ 是迭代初值向量。

例 5.48 求下列方程组在(1,1,1)附近的解并对结果进行验证。

$$\begin{cases} \sin x + y + z^2 e^x = 0 \\ x + yz = 0 \\ xyz = 0 \end{cases}$$

首先建立方程的函数文件 fxyz1.m：

```
function F = F(X)
x = X(1);y = X(2);z = X(3);
F(1) = sin(x) + y + z^2 * exp(x);
F(2) = x + y * z;
F(3) = x * y * z;
```

在 MATLAB 命令窗口，输入命令：

```
X = fsolve('fxyz1',[1,1,1])          %求解 X 的 3 个分量 x、y、z
X =
  0.0015     -0.0136     0.1099
Y = fxyz1(X)                          %检验所求结果 X 是否满足原方程组
Y =
  1.0e-005  *
  0.0004     -0.0003     -0.2235
norm(Y)                               %求 Y 向量的模
ans =
    2.2353e-006
```

Y 向量的模接近 0，说明结果正确。本例要求求解方程组 $F(X)=[0;0;0]$(列向量)，实际上是求出了方程组 $F(X)=Y'$(Y 的转置)的解，当 $\|Y\|\approx 0$ 时，认为后者的根就是所需求解的根。

例 5.49 求圆

$$x^2+y^2+z^2=9$$

和直线

$$\begin{cases} 3x+5y+6z=0 \\ x-3y-6z-1=0 \end{cases}$$

的两个交点。

该问题即为求解方程组

$$\begin{cases} x^2+y^2+z^2-9 = 0 \\ 3x+5y+6z = 0 \\ x-3y-6z-1 = 0 \end{cases}$$

应用函数 fsolve 求解方程组时必须先估计出方程组的根的大致范围。所给直线的方向数是(-12,24,-14),故其与球心在坐标原点的球面的交点大致是(-1,1,-1)和(1,-1,1)。以这两点作为迭代初值。

建立方程组函数文件 fxyz2.m:

```
function F = F(X)
x = X(1);y = X(2);z = X(3);
F(1) = x^2 + y^2 + z^2 - 9;
F(2) = 3 * x + 5 * y + 6 * z;
F(3) = x - 3 * y - 6 * z - 1;
```

在 MATLAB 命令窗口,输入命令:

```
X1 = fsolve('fxyz2',[-1,1,-1])          %求直线与球面的第一个交点
X1 =
   -0.9508     2.4016     -1.5259
X2 = fsolve('fxyz2',[1,-1,1])           %求直线与球面的第二个交点
X2 =
    1.4180     -2.3362     1.2378
```

5.9 稀疏矩阵

在科学研究和工程计算中,所遇到的矩阵往往都是很大的,经常是几百阶甚至更大,并且,其中的大部分元素都为0,非0元素很少,这样的矩阵称为稀疏矩阵。处理稀疏矩阵时,当然也可以不考虑矩阵的稀疏特征,将其和普通矩阵同样处理,但这样做会浪费大量的存储空间,并由此带来访问大量0元素时的时间消耗。计算数学中,专门研究了一套对稀疏矩阵的存储与处理方法。本节先介绍这些方法的基本内容和 MATLAB 中对这些方法的支持,随后以两个示例帮助读者更进一步理解这些内容。

5.9.1 矩阵存储方式

1. 矩阵的完全存储模式

MATLAB 中,矩阵一般是按列存储的,这种存储方式的特征是只存储矩阵元素本身,不存储元素所在的行列位置,每个元素的行列位置由其元素的存储位置决定,这种存储方式称为完全存储方式。完全存储方式没有辅助存储空间开销,直观简洁,但对于具有稀疏特征的矩阵,由于存储了大量的0元素,降低了其对存储空间的利用率。

2. 稀疏矩阵的存储方式

在稀疏存储方式中,一个非0元素用3个数据存储:元素、元素所在的行、元素所在的列。当一个稀疏矩阵的非0元素少于1/3时,这种存储方式有利于提高存储空间利用率。设

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 2 & 0 & 0 & 7 \end{bmatrix}$$

是具有稀疏特征的矩阵，其顺序存储方式是按列存储一个元素序列，共 12 个元素：

1,0,2,0,5,0,0,0,0,0,0,7

其稀疏存储方式如下，括号内为元素的行列位置，其后面为元素本身。

(1,1),1,(3,1),2,(2,2),5,(3,4),7

稀疏存储方式也是占用 12 个元素空间，当原矩阵更加“稀疏”时，会有效地提高空间利用率

注意，在讲稀疏矩阵时，有两个概念需要分清：一是讲矩阵的 0 元素较多，该矩阵是一个具有稀疏特征的矩阵；二是讲采用稀疏方式存储的矩阵。

5.9.2 稀疏存储方式的产生与转化

1. 将一个完全存储方式转化为稀疏存储方式

函数 B = sparse(A)将矩阵 $\boldsymbol{A}$ 转化为稀疏存储方式的矩阵 $\boldsymbol{B}$，当矩阵 $\boldsymbol{A}$ 是稀疏存储方式时，函数调用格式相当于 $\boldsymbol{B}=\boldsymbol{A}$。

例 5.50 设

$$\boldsymbol{A}=\begin{bmatrix}2&0&0&0&0\\0&0&0&0&0\\0&0&0&5&0\\0&1&0&0&-1\\0&0&0&0&-5\end{bmatrix}$$

将 $\boldsymbol{A}$ 转化为稀疏存储方式。

在 MATLAB 命令窗口，输入命令：

```
A=[2,0,0,0,0;0,0,0,0,0;0,0,0,5,0;0,1,0,0,-1;0,0,0,0,-5];
B = sparse(A)
B =
   (1,1)     2
   (4,2)     1
   (3,4)     5
   (4,5)    -1
   (5,5)    -5
```

$\boldsymbol{B}$ 就是 $\boldsymbol{A}$ 的稀疏存储方式。在运算过程中，$\boldsymbol{B}$ 可以直接参与运算。当参与运算的对象不全是稀疏存储矩阵时，所得结果一般是完全存储形式。

sparse 函数还有其他一些格式，逐一说明如下：

sparse(m,n)　生成一个 $m\times n$ 的所有元素都是 0 的稀疏矩阵。

sparse(u,v,S)　$\boldsymbol{u}$、$\boldsymbol{v}$、$\boldsymbol{S}$ 是 3 个等长的向量。$\boldsymbol{S}$ 是要建立的稀疏矩阵的非 0 元素，$u(i)$、$v(i)$ 分别是 $S(i)$的行和列下标，该函数建立一个 max(u)行、max(v)列并以 $\boldsymbol{S}$ 为稀疏元素的稀疏矩阵。

此外，还有一些和稀疏矩阵操作有关的函数。例如：

[U,V,S] = find(A)　返回矩阵 $\boldsymbol{A}$ 中非 0 元素的下标和元素。这里产生的 U、V、S 可作为 sparse(u,v,s)的参数。

full(A)　返回和稀疏存储矩阵 $\boldsymbol{A}$ 对应的完全存储方式矩阵。

2. 产生一个稀疏矩阵

sparse 函数可以将一个完全存储方式的稀疏矩阵转化为稀疏存储方式，但在实际应用时，如果要创立一个 200×150 的稀疏存储方式的矩阵，按照上述方法，则必须先建立该矩阵的完全存储方式矩阵，然后使用 sparse 函数进行转化，这显然是不可取的。能否只把要建立的稀疏矩阵的非 0 元素及其所在行和列的位置表示出来后由 MATLAB 自己产生其稀疏存储方式呢？这需要使用 spconvert 函数。调用格式为：

```
B = spconvert(A)
```

其中 $\boldsymbol{A}$ 为一个 $m\times3$ 或 $m\times4$ 的矩阵，其每行表示一个非 0 元素，m 是非 0 元素的个数，$\boldsymbol{A}$ 中每个元素的意义是：

(i,1)　第 i 个非 0 元素所在的行。

(i,2)　第 i 个非 0 元素所在的列。

(i,3)　第 i 个非 0 元素值的实部。

(i,4)　第 i 个非 0 元素值的虚部，若矩阵的全部元素都是实数，则无须第四列。

该函数将 $\boldsymbol{A}$ 所描述的一个稀疏矩阵转化为一个稀疏存储矩阵。

例 5.51　将稀疏矩阵

$$\boldsymbol{A}=\begin{bmatrix}2 & 2 & 1\\3 & 1 & -1\\4 & 3 & 3\\5 & 3 & 8\\6 & 6 & 12\end{bmatrix}$$

转化为稀疏存储方式。

在 MATLAB 命令窗口，输入命令：

```
A=[2,2,1;3,1,-1;4,3,3;5,3,8;6,6,12];
B=spconvert(A)
B =
    (3,1)    -1
    (2,2)     1
    (4,3)     3
    (5,3)     8
    (6,6)    12
```

应注意，矩阵 $\boldsymbol{A}$ 并非稀疏存储矩阵，只有 $\boldsymbol{B}$ 才是稀疏存储矩阵，为证明这一点，输入命令：

```
full(A)                    %求 A 的完全存储形式
ans =
   2  2    1
   3  1   -1
   4  3    3
   5  3    8
   6  6   12
full(B)                    %求 B 的完全存储形式：
ans =
```

```
     0    0    0    0    0    0
     0    1    0    0    0    0
    -1    0    0    0    0    0
     0    0    3    0    0    0
     0    0    8    0    0    0
     0    0    0    0    0   12
```

3．单位稀疏矩阵的产生

单位矩阵只有对角线元素为1，其他元素都为0，是一种具有稀疏特征的矩阵。函数eye产生一个完全存储方式的单位矩阵，而MATLAB还有一个产生稀疏存储方式的单位矩阵的函数，这就是speye。函数speye(m，n)返回一个 $m \times n$ 的稀疏存储单位矩阵。

5.9.3　稀疏矩阵应用举例

例5.52　求下列齐次方程组

$$
\begin{cases}
x1 - x3 & = 0 \\
x2 - x4 & = 0 \\
-x1 + x3 - x5 & = 0 \\
-x2 + x4 - x6 & = 0 \\
-x3 + x5 & = 0 \\
-x4 + x6 & = 0
\end{cases}
$$

的基础解系。

这个方程组的系数矩阵中0元素较多，这里用稀疏矩阵来解决这个问题。在MATLAB命令窗口，输入命令：

```
S=[1,1,1;1,3,-1;2,2,1;2,4,-1;3,1,-1;3,3,1;3,5,-1;4,2,-1;4,4,1;4,6,-1;5,3,-1;5,5,1;6,4,-1;6,6,1]
```

该命令将产生一个关于系数矩阵的描述，结果是：

```
S =
  1  1   1      %说明第1行第1列元素为1
  1  3  -1      %说明第1行第3列元素为-1
  2  2   1
  2  4  -1
  3  1  -1
  3  3   1
  3  5  -1
  4  2  -1
  4  4   1
  4  6  -1
  5  3  -1
  5  5   1
  6  4  -1
  6  6   1
```

```
A = spconvert(S)      %该命令将对系数矩阵的描述 S 转化为一个稀疏存储矩阵
A =
   (1,1)     1
   (3,1)    -1
   (2,2)     1
   (4,2)    -1
   (1,3)    -1
   (3,3)     1
   (5,3)    -1
   (2,4)    -1
   (4,4)     1
   (6,4)    -1
   (3,5)    -1
   (5,5)     1
   (4,6)    -1
   (6,6)     1
x = null(A,'r')      %该函数将求出原方程组的基础解系
x =
   Empty matrix: 6-by-0
```

说明原方程组没有非 0 解。读者可以计算该系数矩阵的秩,会看到该系数矩阵是一个满秩矩阵。

稀疏矩阵还有很多功能强大的应用,有兴趣的读者可查阅有关资料。

习　题　五

1. 求下列矩阵的主对角元素、上三角阵、下三角阵、逆矩阵、行列式的值、秩、范数、条件数、迹、特征值和特征向量。

(1) $\boldsymbol{A}=\begin{bmatrix}4 & 6 & 0\\ -3 & -5 & 0\\ -3 & -6 & 1\end{bmatrix}$

(2) $\boldsymbol{B}=\begin{bmatrix}1 & 2 & 2\\ 1 & -1 & 1\\ 4 & -12 & 1\end{bmatrix}$

2. 分别用克莱姆(Cramer)法则、矩阵求逆、矩阵除法以及矩阵分解法求下列线性方程组的解。

(1) $\begin{cases}2x-y+3z=9\\ 3x-5y+z=-4\\ 4x-7y+z=5\end{cases}$

(2) $\begin{cases}2x+3y+5z=10\\ 3x+7y+4z=3\\ x+2y+2z=3\end{cases}$

(3) $\begin{cases}2x_1+7x_2+3x_3+x_1 &=6\\3x_1+5x_2+2x_3+2x_4=4\\9x_1+4x_2+x_3+x_4 &=2\end{cases}$

3. 按要求对指定函数进行插值和拟合

表 5.3　特殊角的正弦正切值表

α(度)	0	15	30	45	60	75	90
$\sin\alpha$	0	0.258 8	0.500 0	0.707 1	0.866 0	0.965 9	1.000 0
$\tan\alpha$	0	0.267 9	0.577 4	1.000 0	1.732 0	3.732 0	

表 5.4　1～100 内特殊值的平方根表

N	1	4	9	16	25	36	49	64	81	100
$\sqrt{N}$	1	2	3	4	5	6	7	8	9	10

(1) 按表 5.3 用三次样条方法插值计算 $0°\sim90°$ 内整数点的正弦值和 $0°\sim75°$ 内整数点的正切值，然后用 5 次多项式拟合方法计算相同的函数值，并将两种计算结果进行比较。

(2) 按表 5.4 用三次多项式方法插值计算 1～100 之间整数的平方根。

4. 利用 MATLAB 提供的 randn 函数生成符合正态分布的 10×5 随机矩阵 $\boldsymbol{A}$，进行如下操作：

(1) $\boldsymbol{A}$ 各列元素的均值和标准方差。

(2) $\boldsymbol{A}$ 的最大元素和最小元素。

(3) 分别对 $\boldsymbol{A}$ 的每列元素按升序、每行元素按降序排序。

5. 已知多项式 $P_1(x)=3x+2$，$P_2(x)=5x^2-x+2$，$P_3(x)=x^2-0.5$，求：

(1) $P(x)=P_1(x)P_2(x)P_3(x)$。

(2) $P(x)=0$ 的全部根。

(3) 计算 $x_i=0.2i$，$i=0,1,2,\cdots,10$ 各点上的 $P(x_i)$。

6. 求下列函数在指定点的数值导数。

(1) $f(x)=\sin^2x+\cos^2x$，$x=\frac{\pi}{6},\frac{\pi}{4},\frac{\pi}{3},\frac{\pi}{2}$

(2) $f(x)=\sqrt{x^2+1}$，$x=1,2,3$

7. 求数值积分。

(1) $\int_0^{\pi}\sin^5x\sin5x\,dx$

(2) $\int_{-1}^{1}\frac{1+x^2}{1+x^4}dx$

(3) $\oint_c\sqrt{x^2+y^2}ds$，$c:x^2+y^2=1$。

(4) $\iint\limits_{\Omega}|\cos(x+y)|\,dxdy$，$\Omega:0\leqslant x\leqslant\pi,0\leqslant y\leqslant\pi$。

8. 求下列函数在指定区间的最大值：

(1) $f(x)=\frac{1+x^2}{1+x^4}$，$x\in(0,2)$。

(2) $f(x)=\sin x+\cos x^2$，$x\in(0,\pi)$。

9. 求下列微分方程的数值解。

(1) $\begin{cases} \dfrac{dy}{dx} - 2x = \dfrac{2x}{y} \\ y(1) = 0 \end{cases}$

(2) $\begin{cases} x^2 \dfrac{d^2y}{dx^2} + 4x \dfrac{dy}{dx} + 2y = 0 \\ y(1) = 2 \\ y'(1) = -3 \end{cases}$

10. 求下列非线性方程的数值解。

(1) $x^4 + 3x^3 - 2x^2 + 5x - 10 = 0$

(2) $\begin{cases} x - 2y + z = -1 \\ 3x - 5y + 2z = 2 \\ x^2 + y^2 + z^2 = 49 \end{cases}$

第 6 章　MATLAB 符号计算

符号计算是 MATLAB 的一个重要组成部分。应用符号计算功能,可以直接对抽象的符号对象进行微积分与代数计算,并获得问题的解析结果。MATLAB 中的符号计算功能是以 Maple V 为基础开发的,集成在 MATLAB 的符号运算工具箱(Symbolic Math Toolbox)中。用户必须在 MATLAB 安装时选择安装符号运算工具箱模块,才能运行与符号运算有关的函数。

本章首先介绍符号计算基础,然后介绍符号微积分、级数的符号求和与函数的级数表示、代数方程和微分方程的符号求解等内容。

6.1　符号计算基础

在科学研究和工程实践活动中,除存在大量的数值计算外,还有对符号对象进行的运算。符号计算可以获得比数值计算更一般的结果。MATLAB 中为用户提供了一种符号数据类型,相应的变量和常量称为符号变量和符号常数。用户在定义了符号对象(符号变量、符号常数以及有它们参与的数学表达式)后,可以进行符号对象的运算。

6.1.1　符号对象

进行符号运算前首先要建立符号对象。如前所述,符号对象包括符号变量、符号常数以及符号表达式等。

1. 建立符号变量和符号常数

MATLAB 中提供了 2 个建立符号对象的函数:sym 和 syms,两个函数的用法不同。

(1) sym 函数

sym 函数用来建立单个符号量,例如,a = sym('a')建立符号变量 a,此后,用户可以在表达式中使用变量 a 进行各种运算。读者不要将符号变量 a 和在其他过程中建立的非符号变量 a 混淆。一个非符号变量在参与运算前必须赋值,变量的运算实际上是该变量所对应值的运算,其运算结果是一个和变量类型对应的值。而符号变量参与计算前无须赋值,其结果是一个由参与运算的变量名组成的表达式。为使读者理解这一点,下面看一个实例。

例 6.1　考察符号变量和数值变量的差别。

在 MATLAB 命令窗口,输入命令:

```
a = sym('a');b = sym('b');c = sym('c');d = sym('d');     %定义 4 个符号变量
w = 10;x = 5;y = -8;z = 11;                               %定义 4 个数值变量
A = [a,b;c,d]                                             %建立符号矩阵 A
A =
[a, b]
```

```
[c, d]
B = [w,x;y,z]                                    %建立数值矩阵 B
B =
    10      5
    -8     11
det(A)                                           %计算符号矩阵 A 的行列式
ans =
    a * d - b * c
det(B)                                           %计算数值矩阵 B 的行列式
ans =
    150
```

应用 sym 函数还可以定义符号常数,使用符号常数进行代数运算时与数值常数进行的运算有些差别。

例 6.2　比较符号常数与数值在代数运算时的差别。

在 MATLAB 命令窗口,输入命令:

```
pi1 = sym('pi');k1 = sym('8');k2 = sym('2');k3 = sym('3');   %定义符号变量
pi2 = pi;r1 = 8;r2 = 2;r3 = 3;                               %定义数值变量
sin(pi1/3)                                                   %计算符号表达式值
ans =
1/2 * 3^(1/2)
sin(pi2/3)                                                   % 计算数值表达式值
ans =
    0.8660
sqrt(k1)                                                     %计算符号表达式值
ans =
2 * 2^(1/2)
sqrt(r1)                                                     %计算数值表达式值
ans =
    2.8284
sqrt(k3 + sqrt(k2))                                          %计算符号表达式值
ans =
(3 + 2^(1/2))^(1/2)
sqrt(r3 + sqrt(r2))                                          %计算数值表达式值
ans =
    2.1010
```

从该例可以看出,用符号常数进行计算更像在进行数学演算,所得到的结果是精确的数学表达式,而数值计算将结果近似为一个有限小数。

在例 6.1 和 6.2 中,读者可以看到,函数 sym 一次只能定义一个符号变量,使用不方便。MATLAB 提供了另一个函数 syms,一次可以定义多个符号变量。

(2) syms 函数

syms 函数的一般调用格式为:

```
syms var1 var2 … varn
```

函数定义符号变量 *var*1, *var*2, …, *varn* 等。用这种格式定义符号变量时不要在变量名上加字符分界符('),变量间用空格而不要用逗号分隔。例如,用 syms 函数定义 4 个符号变量 *a*, *b*, *c*, *d*,命令如下:

```
syms a b c d
```

2. 建立符号表达式

含有有符号量的表达式称为符号表达式。符号表达式包括一般代数式、符号方程、符号矩阵和抽象函数等。

(1) 建立代数式

建立符号表达式的方法有两种:一是直接用 sym 函数建立符号表达式,二是使用已经定义的符号变量组成符号表达式。

例 6.3 用两种方法建立符号表达式:$3x^2+5y+2xy+6$。

在 MATLAB 窗口,输入命令:

```
U = sym('3 * x^2 + 5 * y + 2 * x * y + 6')          %定义符号表达式 U,此时不需定义变量 x、y
U =
3 * x^2 + 5 * y + 2 * x * y + 6
syms x y;                                            %建立符号变量 x、y
V = 3 * x^2 + 5 * y + 2 * x * y + 6                  %定义符号表达式 V
ans =
3 * x^2 + 5 * y + 2 * x * y + 6
2 * U - V + 6                                        %求符号表达式的值
ans =
3 * x^2 + 5 * y + 5 * x * y + 12
```

(2) 建立符号方程

利用 sym 函数可以建立符号方程。例如,y = sym('a * x^2 + bx + c = 0')建立一个符号方程 $ax^2+bx+c=0$。因为符号方程中含有等号,所以不能采用赋值的方法来建立符号方程,否则方程中的等号和命令中的赋值号会混淆。

(3) 建立符号矩阵

在例 6.1 中已经建立了一个符号矩阵 **A**,这里采用的是直接输入法。也可以由一些小的符号矩阵来构成一个大的符号矩阵。

例 6.4 计算 3 阶范得蒙德(Vandermonde)矩阵行列式的值。设 **A** 是一个由符号变量 *a*, *b*, *c* 确定的范得蒙德矩阵。

$$\boldsymbol{A}=\begin{bmatrix}1 & 1 & 1\\ a & b & c\\ a^2 & b^2 & c^2\end{bmatrix}$$

用 vander 函数可以建立一个范得蒙德数值矩阵,但对一组符号变量,不能用 vander 函数,必须用其他的方法建立矩阵。命令如下:

```
syms a b c;
U = [a,b,c];
```

```
A = [[1,1,1];U;U.^2]                          %建立范得蒙德符号矩阵
A =
[   1,   1,   1]
[   a,   b,   c]
[a^2, b^2, c^2]
det(A)                                        %计算 A 的行列式值
ans =
b * c^2 - c * b^2 - a * c^2 + a * b^2 + a^2 * c - a^2 * b
```

(4) 建立抽象函数

符号表达式和符号矩阵中,尽管其表达式本身的值未确定,但函数关系是确定的,所谓抽象函数,是函数关系也未确定的一种函数。例如,函数 $f(x,y)$ 只知道函数 f 依赖于变量 x 和 y,但并未确定具体的依赖关系,这是一个抽象函数。在高等数学中所研究的多是这一类函数。

例 6.5　建立 x,y 的一般二元函数。

在 MATLAB 命令窗口,输入命令:

```
syms x y;
f = sym('f(x,y)');
```

6.1.2　基本的符号运算

1. 符号表达式运算

(1) 符号表达式的四则运算

符号表达式的四则运算和其他表达式的运算并无不同,但要注意,其运算结果依然是一个符号表达式。且在很多时候,MATLAB 并未将结果化为最简形式。

例 6.6　符号表达式的四则运算示例。

在 MATLAB 命令窗口,输入命令:

```
syms x y z;
f = 2 * x + x^2 * x - 5 * x + x^3             %符号表达式的结果为最简形式
f =
-3 * x + 2 * x^3
f = 2 * x/(5 * x)                             %符号表达式的结果为最简形式
f =
2/5
f = (x + y) * (x - y)                         %符号表达式的结果不是 x^2 - y^2,而是(x + y) * (x - y)
f =
(x + y) * (x - y)
```

(2) 因式分解与展开

MATLAB 提供了几个专用于符号表达式运算的函数,它们既可用于一般符号表达式,也可用于符号矩阵,下面简要介绍这几个函数的功能和用法。

factor(S)　对 S 分解因式,S 是符号表达式或符号矩阵。

expand(S)　对 S 进行展开,S 是符号表达式或符号矩阵。

collect(S)　对 S 合并同类项,S 是符号表达式或符号矩阵。

collect(S,v) 对 S 按变量 v 合并同类项，S 是符号表达式或符号矩阵。

例 6.7 对下列符号矩阵 A 的每个元素分解因式。

$$A=\begin{bmatrix}2a^2b^3x^2-4ab^4x^3+10ab^6x^4 & 3xy-5x^2\\ 4 & a^3-b^3\end{bmatrix}$$

符号矩阵 A 涉及 a,b,x,y 等 4 个符号变量，首先定义符号变量及符号矩阵，然后应用 factor 函数分解因子。命令如下：

```
syms a b x y;
A = [2 * a^2 * b^3 * x^2 - 4 * a * b^4 * x^3 + 10 * a * b^6 * x^4,3 * x * y - 5 * x^2;4,a^3 - b^3];
factor(A)                    %对 A 的每个元素分解因式
ans =
[2 * a * x^2 * b^3 * (5 * x^2 * b^3 - 2 * x * b + a),          - x * ( - 3 * y + 5 * x)]
[                    4,                  (a - b) * (a^2 + a * b + b^2)]
```

例 6.8 计算表达式 S 的值。

$$S=(-7x^2-8y^2)\times(-x^2+3y^2)$$

命令如下：

```
syms x y;
s = ( - 7 * x^2 - 8 * y^2) * ( - x^2 + 3 * y^2);
expand(s)                                  %对 s 展开
ans =
7 * x^4 - 13 * x^2 * y^2 - 24 * y^4
collect(s,x)                               %对 s 按变量 x 合并同类项(无同类项)
ans =
7 * x^4 - 13 * x^2 * y^2 - 24 * y^4
factor(ans)                                %对 ans 分解因式
ans =
(7 * x^2 + 8 * y^2) * (x^2 - 3 * y^2)
```

(3) 表达式化简

MATLAB 提供的对符号表达式化简的函数有：

simplify(S) 应用函数规则对 S 进行化简。

simple(S) 调用 MATLAB 的其他函数对表达式进行综合化简，并显示化简过程。

例 6.9 化简

$$s=(x^2+y^2)^2+(x^2-y^2)^2$$

命令如下：

```
syms x y;
s = (x^2 + y^2)^2 + (x^2 - y^2)^2;
simple(s)          %MATLAB 自动调用多种函数对 s 进行化简，并显示每步结果
simplify:          %这是 simple 函数自动调用的函数，并非由用户键入。下同
2 * x^4 + 2 * y^4
radsimp:
2 * x^4 + 2 * y^4
```

```
combine(trig):
2 * x^4 + 2 * y^4
factor:
2 * x^4 + 2 * y^4
expand:
2 * x^4 + 2 * y^4
combine:
(x^2 + y^2)^2 + (x^2 - y^2)^2
convert(exp):
(x^2 + y^2)^2 + (x^2 - y^2)^2
convert(sincos):
(x^2 + y^2)^2 + (x^2 - y^2)^2
convert(tan):
(x^2 + y^2)^2 + (x^2 - y^2)^2
collect(x):
2 * x^4 + 2 * y^4
ans =
2 * x^4 + 2 * y^4
```

simple 函数试探使用各种函数化简,其化简过程是机械性的,其中有些步骤可以省略。

2. 符号矩阵运算

上述用于一般符号表达式的函数都可用于符号矩阵,但应注意这些函数用于符号矩阵时,是分别作用于矩阵的每一个元素。除上面介绍的函数外,MATLAB 还有一些专用于符号矩阵的函数,这些函数作用于单个的数据无意义。

transpose(S) 返回 S 矩阵的转置矩阵。

determ(S) 返回 S 矩阵的行列式值。

colspace(S) 返回 S 矩阵列空间的基。

[Q,D] = eigensys(S) $\boldsymbol{Q}$ 返回 $\boldsymbol{S}$ 矩阵的特征向量,$\boldsymbol{D}$ 返回 $\boldsymbol{S}$ 矩阵的特征值。

其实,第5章曾介绍过的许多应用于数值矩阵的函数,如 diag、triu、tril、inv、det、rank、eig 等,也可直接应用于符号矩阵。

6.1.3 符号表达式中变量的确定

MATLAB 中的符号可以表示符号变量和符号常数。findsym 可以帮助用户查找一个符号表达式中的符号变量。该函数的调用格式为:

```
findsym(S,n)
```

函数返回符号表达式 S 中的 n 个符号变量,若没有指定 n,则返回 S 中的全部符号变量。例如

```
syms x a y z b;                              %定义5个符号变量
s1 = 3 * x + y;s2 = a * y + b;               %定义符号表达式
findsym(s1),findsym(s2,2),findsym(5 * x + 2)
ans =
```

```
x, y
ans =
y,b
ans =
x
c = sym('3');                          %定义符号常数 c
findsym(a * x + b * y + c)             %符号常数 c 不在结果中出现
ans =
a, b, x, y
```

在求函数的极限、导数和积分时，如果用户没有明确指定自变量，MATLAB 将按缺省原则确定主变量并对其进行相应微积分运算。可用 findsym(S,1)查找系统的缺省变量，事实上，MATLAB 按离字符'x'最近原则确定缺省变量。例如：

```
findsym(a * y + b * w,1)
ans =
y
findsym(a * z + b * w,1)
ans =
w
```

6.2 符号导数及其应用

学习 MATLAB 符号计算的目的在于借助于 MATLAB 的强大功能解决实际应用中遇到的问题，减少用户在数学上的大量推理和计算。极限与导数是高等数学的基础，现在就研究如何应用 MATLAB 解决求函数极限和导数的问题。

6.2.1 函数的极限

MATLAB 中求函数极限的函数是 limit，可用来求函数在指定点的极限值和其左右极限值。limit 函数的调用格式为：

limit(f,x,a)

函数求极限 $\lim\limits_{x \to a} f(x)$。格式中的参数 a，x 可以省略。当省略参数 x 时，MATLAB 按缺省原则确定求极限的变量。可使用函数 findsym(f,1)帮助查找符号表达式 f 的缺省变量。当省略参数 a 时，MATLAB 默认变量趋近于 0，即相当于 $a = 0$ 的情况。

limit 函数的另一种功能是求单边极限，其调用格式为：

limit(f,x,a,'right') 或 limit(f,x,a,'left')

分别求极限 $\lim\limits_{x \to a^+} f(x)$ 和 $\lim\limits_{x \to a^-} f(x)$。参数 x 和 a 的取值规则同前。

例 6.10 求下列极限。

(1) $\lim\limits_{x \to a} \dfrac{\sqrt[m]{x} - \sqrt[m]{a}}{x - a}$　　(2) $\lim\limits_{x \to 0} \dfrac{\sin(x + a) - \sin(x - a)}{x}$

(3) $\lim\limits_{x \to +\infty} x(\sqrt{x^2+1}-x)$　　(4) $\lim\limits_{x \to a^+} \frac{\sqrt{x}-\sqrt{a}+\sqrt{x-a}}{\sqrt{x^2-a^2}}$

在 MATLAB 命令窗口，输入命令：

```
syms a m x;
f = (x^(1/m) - a^(1/m))/(x - a);
limit(f,x,a)                          %求极限(1)
ans =
a^(1/m)/a/m
f = (sin(a + x) - sin(a - x))/x;
limit(f)                              %求极限(2)
ans =
2 * cos(a)
f = x * (sqrt(x^2 + 1) - x);
limit(f,inf)                          %求 f 函数在 x→∞(包括 + ∞和 - ∞)处的极限
ans =
NaN
limit(f,x,inf,'left')                 %求极限(3)
ans =
1/2
f = (sqrt(x) - sqrt(a) - sqrt(x - a))/sqrt(x * x - a * a);
limit(f,x,a,'right')                  %求极限(4)
ans =
 - 1/2 * 2^(1/2)/a^(1/2)
```

6.2.2 符号函数求导及其应用

导数的概念是读者已经熟悉的，它是高等数学的基础，是现代数学的一个基本工具，这一节讨论怎样应用 MATLAB 求函数的一阶导数、高阶导数以及求多变量函数的导数。

MATLAB 中的求导的函数为：

```
diff(f,x,n)
```

diff 函数求函数 f 对变量 x 的 n 阶导数。参数 x 的用法同求极限函数 limit，可以缺省，缺省值与 limit 相同，n 的缺省值是 1。

例 6.11　求下列函数的导数。

(1) $y=\sqrt{1+e^x}$，求 y'。　　(2) $y=x\cos(x)$，求 y''、y'''。

(3) $\begin{cases} x = a\cos t \\ y = b\sin t \end{cases}$，求 y'_x、y''_x。　　(4) $z=\frac{xe^y}{y^2}$，求 z'_x、z'_y。

(5) $z=f(x,y)$ 由方程 $x^2+y^2+z^2=a^2$ 定义，求 z'_x、z'_y。

命令如下：

```
syms a b t x y z;
f = sqrt(1 + exp(x));
diff(f)                               %求(1)。未指定求导变量和阶数，按缺省规则处理
```

```
ans =
1/2/(1 + exp(x))^(1/2) * exp(x)
f = x * cos(x);
diff(f,x,2)                                    %求(2)。求 f 对 x 的二阶导数
ans =
-2 * sin(x) - x * cos(x)
diff(f,x,3)                                    %求(2)。求 f 对 x 的三阶导数
ans =
-3 * cos(x) + x * sin(x)
f1 = a * cos(t); f2 = b * sin(t);
diff(f2)/diff(f1)                              %求(3)。按参数方程求导公式求 y 对 x 的导数
ans =
-b * cos(t)/a/sin(t)
(diff(f1) * diff(f2,2) - diff(f1,2) * diff(f2))/(diff(f1))^3      %求(3)。求 y 对 x 的二阶导数
ans =
-(a * sin(t)^2 * b + a * cos(t)^2 * b)/a^3/sin(t)^3
f = x * exp(y)/y^2;
diff(f,x)                                      %求(4)。z 对 x 的偏导数
ans =
exp(y)/y^2
diff(f,y)                                      %求(4)。z 对 y 的偏导数
ans =
x * exp(y)/y^2 - 2 * x * exp(y)/y^3
f = x^2 + y^2 + z^2 - a^2;
zx = - diff(f,x)/diff(f,z)                     %求(5)。按隐函数求导公式求 z 对 x 的偏导数
zx =
-x/z
zy = - diff(f,y)/diff(f,z)                     %求(5)。按隐函数求导公式求 z 对 y 的偏导数
zy =
-y/z
```

解题过程中未对结果化简处理，读者可用 6.1.2 节中的有关函数进一步简化有些导函数的结果。

例 6.12　求曲线 $y = x^3 + 3x - 2$ 上与直线 $y = 4x - 1$ 平行的切线的切点。

依题意，即求曲线哪一点的导数值为 4。命令如下：

```
x = sym('x');
y = x^3 + 3 * x - 2;                           %定义曲线函数
f = diff(y);                                   %对曲线求导数
g = f - 4;
solve(g)                                       %求方程 f - 4 = 0 的根，即求曲线何处的导数为 4
ans =
[1/3 * 3^(1/2)]
```

```
[-1/3*3^(1/2)]
```

结果表明，在 $x=\frac{\sqrt{3}}{3}$ 和 $x=-\frac{\sqrt{3}}{3}$ 处的切线和指定直线平行。

从这个例子可以看到，MATLAB 尽管可以进行很多繁杂的计算，但应用 MATLAB 解决实际问题时，解题者自身的分析过程也是必不可少的。

6.3 符号积分

在第 5 章介绍了求定积分的数值方法。数值方法最大的好处就是"可行性"，不管被积函数的形式如何、复杂程度怎样，采用数值积分法总可以求得一个结果，尽管这种结果大部分情况下是近似的。数值方法最大的缺点就是不能获得解析解。这一节讨论符号函数的积分，这里讨论的方法可以获得和高等数学中一样的解析结果。积分算法是非结构性的，许多函数的原函数存在，但不可用有限解析表达式表示，即使可以求积的函数，其求积过程也可能很复杂，但利用 MATLAB 求积分就非常容易。

6.3.1 不定积分

求函数的不定积分是求导数的逆运算。在 MATLAB 中，求不定积分的函数是 int，其调用格式为：

```
int(f,x)
```

int 函数求函数 f 对变量 x 的不定积分。参数 x 可以缺省，缺省原则与 diff 函数相同。

例 6.13　求下列不定积分。

(1) $\int(3-x^2)^3\mathrm{d}x$

(2) $\int\sqrt{x^3+x^4}\mathrm{d}x$

命令如下：

```
x=sym('x');
f=(3-x^2)^3;
int(f)                              %求不定积分(1)
ans =
-1/7*x^7+9/5*x^5-9*x^3+27*x
f=sqrt(x^3+x^4);
int(f)                              %求不定积分(2)
ans =
-1/48*(x^3+x^4)^(1/2)*(-16*(x^2+x)^(3/2)+12*(x^2+x)^(1/2)*x+6*(x^2+x)^(1/2)-3*log
(1/2+x+(x^2+x)^(1/2)))/x/((x+1)*x)^(1/2)
g=simple(ans)                       %调用 simple 函数对结果化简
g =
1/3*x^(5/2)*(x+1)^(1/2)+1/12*x^(3/2)*(x+1)^(1/2)-1/8*x^(1/2)*(x+1)^(1/2)+1/16*
```

```
log(1/2 + x + (x^2 + x)^(1/2))
```

上述结果还可进一步化简，这也说明 MATLAB 还有需进一步完善的地方。

6.3.2 符号函数的定积分

定积分在实际工作中有广泛的应用。在 MATLAB 中，定积分的计算使用函数：

```
int(f,x,a,b)
```

该函数计算函数 f 在区间 $[a,b]$ 或 (a,b) 上的定积分。参数 x 的意义与求不定积分函数相同。a 和 b 可以是两个具体的数，也可以是一个符号表达式，还可以是无穷(inf)。当函数 f 关于变量 x 在闭区间 $[a,b]$ 上可积时，函数返回一个定积分结果。当 a,b 中有一个是 inf 时，函数返回一个广义积分。当 a,b 中有一个是符号表达式时，函数返回一个符号函数。

例 6.14 求下列定积分，并对(4)与数值积分结果进行比较。

(1) $\int_1^2 |1-x| \mathrm{d}x$

(2) $\int_{-\infty}^{+\infty} \frac{\mathrm{d}x}{1+x^2}$

(3) $\int_2^{\sin t} 4tx \mathrm{d}x$

(4) $\int_2^3 \frac{x^3 \mathrm{d}x}{(x-1)^{100}}$

命令如下：

```
x = sym('x');t = sym('t');
int(abs(1 - x),1,2)                         %求定积分(1)
ans =
1/2
f = 1/(1 + x^2);
int(f, - inf,inf)                           %求定积分(2)
ans =
pi
int(4 * t * x,x,2,sin(t))                   %求定积分(3)
ans =
2 * t * (sin(t)^2 - 4)
f = x^3/(x - 1)^100;
I = int(f,2,3)                              %用符号积分的方法求定积分(4)
I =
97893129180187301565519001875382615/1192978373971185320372138406360121344
double(I)                                   %将上述符号结果转换为数值
ans =
    0.0821
```

为应用数值方法求定积分(4)，先建立一个函数文件 fxx.m：

```
function f = f(x)
f = x.^3./(x - 1).^100;                     %注意应使用点运算
```

再在 MATLAB 命令窗口，输入命令：

```
I1 = quad8('fxx',2,3,1e-6)                    %用数值积分方法求定积分(4)
I1 =
    0.0821
```

计算结果表明两种计算方法的结果完全一致。

例 6.15　求椭球的体积。

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$$

用平面 $z = z_0(z_0 \leqslant c)$ 去截取上述椭球时，其相交线是一个椭圆，该椭圆在 xy 平面投影的面积是 $S(z) = \frac{\pi ab(c^2 - z^2)}{c^2}$，椭球的体积 $V = \int_{-c}^{c} S(z)\mathrm{d}z$。

命令如下：

```
syms a b c z;
f = pi * a * b * (c^2 - z^2)/c^2;
V = int(f,z, - c,c)
V =
4/3 * pi * a * b * c
```

例 6.16　轴的长度为 10 m，若该轴的线密度计算公式是 $f(x) = 6 + 0.3x$ (kg/s)(其中 x 为距轴的端点的距离)，求轴的质量。

轴的质量应是轴密度函数沿整个轴的积分。下面应用数值积分和符号函数两种方式计算轴的质量。

(1) 符号函数积分。在 MATLAB 命令窗口，输入命令：

```
syms x;
f = 6 + 0.3 * x;
m = int(f,0,10)
m =
75
```

(2) 数值积分。

先建立一个函数文件 fx.m：

```
function fx = fx(x)
fx = 6 + 0.3 * x;
```

再在 MATLAB 命令窗口，输入命令：

```
m = quad('fx',0,10,1e-6)
m =
75
```

由于本例很简单，两种计算方法的结果完全一样，大部分情况下，两者会存在极小的差别，一般符号函数积分的结果是准确的。但读者不要认为有了符号函数积分后，数值积分就毫无用处，在不能用有限形式表示一个函数的原函数，或被积函数只有列表表示形式时，数值积分方法是符号函数积分无法替代的。

例 6.17　求空间曲线 c 从点(0,0,0)到点(3,3,2)的长度。设曲线 c 的方程是：

$$\begin{cases} x = 3t \\ y = 3t^2 \\ z = 2t^3 \end{cases}$$

求曲线 c 的长度是曲线一型积分问题,按公式可转化为定积分问题。这里曲线的始终点对应参数 $t=0$ 和 $t=1$,曲线积分转化为定积分的公式是:

$$\int_c f(x,y,z) = \int_0^1 f(x(t),y(t),z(t))\sqrt{(x'(t))^2+(y'(t))^2+(z'(t))^2}\,\mathrm{d}t$$

计算曲线长度时,被积函数 $f=1$。

命令如下:

```
syms t;
x = 3 * t;y = 3 * t^2;z = 2 * t^3;
f = diff([x,y,z],t)                                  %求 x,y,z 对参数 t 的导数
f =
[      3,  6 * t, 6 * t^2]
g = sqrt(f * f')                                     %计算一型积分公式中的根式部分
g =
3 * (1 + 4 * t * conj(t) + 4 * t^2 * conj(t)^2)^(1/2)
l = int(g,t,0,1)                                     %计算曲线 c 的长度
l =
5
```

6.3.3 积分变换

所谓积分变换,就是通过积分运算,把一个函数 f 变成另外一个函数 F,变换过程是

$$F(t) = \int_a^b f(x)K(x,t)\mathrm{d}x$$

其中二元函数 $K(x,t)$称为变换的核,变换的核决定了变换的不同名称。f 称为变换 F 的原函数,而 $F(t)$称为 $f(x)$的像函数。在一定条件下,像函数 $F(t)$和原函数 $f(x)$间是一一对应,可以相互转化的。

积分变换的一项基本应用是求解微分方程,求解过程是基于这样一种想法:假如不容易从原方程直接求得解 f,则对原方程进行变换,如果能从变换以后的方程中求得解 F,则对 F 进行逆变换,即可求得原方程的解 f,当然,在选择变换的核时,应使得变换以后的方程比原方程容易求解。

常见的积分变换有傅立叶变换、拉普拉斯变换和 Z 变换。

1. 傅立叶(Fourier)变换

当积分变换的核

$K(x,t) = \mathrm{e}^{-\mathrm{i}tx}$(其中 i 为虚数单位)

称积分变换

$$F(t) = \int_{-\infty}^{+\infty} f(x)\mathrm{e}^{-\mathrm{i}tx}\mathrm{d}x$$

为傅立叶变换,其逆变换为

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(t)e^{ixt}dt$$

在 MATLAB 中,进行傅立叶变换的函数是:

fourier(fx,x,t)　求函数 $f(x)$的傅立叶像函数 $F(t)$。

ifourier(Fw,t,x)　求傅立叶像函数 $F(t)$的原函数 $f(x)$。

例 6.18　求函数 $y=|x|$的傅立叶变换及其逆变换。

命令如下:

```
syms x t;
y = abs(x);
Ft = fourier(y,x,t)                          %求 y 的傅立叶变换
Ft =
-2/t^2
fx = ifourier(Ft,t,x)                        %求 Ft 的傅立叶逆变换
fx =
x * (Heaviside(x) - Heaviside( - x))
```

结果中的 Heaviside 是 Maple 函数库中的一个函数,数学上称为单位阶跃函数。其定义是

$$\text{Heaviside}(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

不能在 MATLAB 中调用 Heaviside 函数。

2. 拉普拉斯(Laplace)变换

拉普拉斯变换在微分方程、信号分析及自动控制方面有广泛的应用。当积分变换的核为

$$K(x,t) = e^{-xt}$$

称积分变换

$$F(t) = \int_0^{+\infty} f(x)e^{-xt}dx$$

为拉普拉斯变换,其逆变换为

$$f(x) = \int_0^{+\infty} F(t)e^{xt}dt$$

在 MATLAB 中,进行拉普拉斯变换的函数是:

laplace(fx,x,t)　求函数 $f(x)$的拉普拉斯像函数 $F(t)$。

ilaplace(Fw,t,x)　求拉普拉斯像函数 $F(t)$的原函数 $f(x)$。

例 6.19　计算 $y=x^2$ 的拉普拉斯变换及其逆变换。

命令如下:

```
x = sym('x');y = x^2;
Ft = laplace(y,x,t)                          %对函数 y 进行拉普拉斯变换
Ft =
2/t^3
fx = ilaplace(Ft,t,x)                        %对函数 Ft 进行拉普拉斯逆变换
fx =
```

```
x^2
```

3. Z 变换

当函数 $f(x)$ 呈现为一个离散的数列 $f(n)$ 时，称变换

$$F(z) = \sum_{n=0}^{+\infty} f(n) e^{-n}$$

为 Z 变换，其逆变换为

$$f(n) = \frac{1}{2i\pi} \oint_{\Gamma} F(z) z^{n-1} dz$$

其中 i 为虚数单位。

对数列 $f(n)$ 进行 Z 变换的 MATLAB 函数是：

ztrans(fn,n,z)　求 fn 的 Z 变换像函数 $F(z)$

iztrans(Fz,z,n)　求 Fz 的 Z 变换原函数 $f(n)$

例 6.20　求数列 $fn = e^{-n}$ 的 Z 变换及其逆变换。

命令如下：

```
syms n z
fn = exp(-n);
Fz = ztrans(fn,n,z)                         % 求 fn 的 Z 变换
Fz =
z/exp(-1)/(z/exp(-1) - 1)
f = iztrans(Fz,z,n)                         % 求 Fz 的逆 Z 变换
f =
exp(-1)^n
```

4. 积分变换的应用

下面举例说明拉普拉斯变换在求解微分方程中的应用。

例 6.21　用拉普拉斯方法解微分方程。

$$y'' - 2y' - 3y = 4e^{2x}$$
$$y(0) = 2, y'(0) = 8$$

用拉普拉斯方法解微分方程的基本思想是对方程两端同时进行拉普拉斯变换，并利用拉普拉斯变换的性质求出未知函数 y 的像函数 $L[y]$，然后对 $L[y]$ 进行逆变换。

命令如下：

```
syms x t;
y = sym('y(x)');                            % 定义未知函数为一个抽象函数
Ft1 = laplace(diff(y,2) - 2 * diff(y) - 3 * y,x,t)   % 对方程左端进行拉普拉斯变换
Ft1 =
t * (t * laplace(y(x),x,t) - y(0)) - D(y)(0) - 2 * t * laplace(y(x),x,t) + 2 * y(0) - 3 * laplace(y(x),x,t)
Ft2 = laplace(4 * exp(2 * x),x,t)           % 对方程右端进行拉普拉斯变换
Ft2 =
4/(t - 2)
```

令 Ft1 = Ft2，并将初值条件 y(0) = 2，Dy(0) = 8 代人，得

```
(t * t - 2t - 3) * laplace(y(x),x,t) - 2t - 4 = 4/(t - 2)
```

这是一个关于 laplace(y(x),x,t) 的代数方程，解这个方程，得

```
laplace(y(x),x,t) = (2t + 4 + 4/(t - 2))/(t * t - 2t - 3)
```

对 laplace(y(x),x,t) 求拉普拉斯逆变换，即得到 y(x)。命令如下：

```
y = ilaplace((2 * t + 4 + 4/(t - 2))/(t * t - 2 * t - 3),t,x)        % 进行拉普拉斯逆变换
y =
- 4/3 * exp(2 * x) - 1/6 * exp( - x) + 7/2 * exp(3 * x)
```

通过拉普拉斯变换及其逆变换，将一个微分方程的求解问题转化为一个代数方程的求解问题，这种思想读者应认真体会。另外，本例中将未知函数 y 定义为一个关于 x 的抽象函数而不是一般的字符变量，这也应引起读者注意。

6.4　级　　数

数值级数与函数级数是高等数学的重点研究内容，级数也是物理学以及其他工程技术学科的重要理论基础和分析工具。这一节讨论 MATLAB 有关级数的函数。

6.4.1　级数的符号求和

在第 5 章讨论过有限级数求和的函数 sum，sum 处理的级数是以一个向量形式表示的，并且只能是有穷级数，对于下面的等比无穷级数求和

$$S = \sum_{n=1}^{+\infty} \frac{1}{n^2}$$

sum 是无能为力的。求级数的和 S 需要符号表达式求和函数 symsum，调用格式为

```
symsum(a,n,n0,nn)
```

其中 a 表示一个级数的通项，是一个符号表达式。n 是求和变量（n 可以省略，省略时使用系统的缺省变量），$n0$ 和 nn 是求和的开始项和末项。函数的功能是求和：$S = a(n0) + \cdots + a(nn)$。

例 6.22　求下列级数之和。

(1) $s1 = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \cdots + \frac{1}{n^2} + \cdots$

(2) $s2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + (-1)^{n+1}\frac{1}{n} + \cdots$

(3) $s3 = x + 2x^2 + 3x^3 + \cdots + nx^n + \cdots$

(4) $s4 = 1 + 4 + 9 + 16 + \cdots 10000$

命令如下：

```
n = sym('n');
s1 = symsum(1/n^2,n,1,inf)                 %求 s1
s1 =
1/6 * pi^2
s2 = symsum(( - 1)^(n + 1)/n,1,inf)        %求 s2。未指定求和变量，缺省为 n
s2 =
log(2)
```

```
s3 = symsum(n * x^n,n,1,inf)                    %求 s3。此处的求和变量 n 不能省略。
s3 =
x/(x-1)^2
s4 = symsum(n^2,1,100)                          %求 s4。计算有限级数的和
s4 =
338350
```

6.4.2 函数的泰勒级数

泰勒(Taylor)级数可以将一个任意函数表示为一个幂级数,并且,在许多情况下,只需要取幂级数的前有限项对于大多数工程应用问题已经足够。在高等数学中讨论了怎样将一个任意(当然也应该满足泰勒条件)的函数在某点 x_0 附近表示为幂级数的方法,MATLAB 中提供了将函数展开为幂级数的函数 taylor,其调用格式为:

```
taylor(f,v,n,a)
```

该函数将函数 f 按变量 v 展开为泰勒级数,展开到第 n 项(即变量 v 的 $n-1$ 次幂)为止,n 的缺省值为 6。v 的缺省值与 diff 函数相同。参数 a 指定将函数 f 在自变量 $v=a$ 处展开,a 的缺省值是 0。

例 6.23 求下列函数在指定点的泰勒展开式。

(1) $\dfrac{1+x+x^2}{1-x+x^2}$,展开到含 x^4 的项。

(2) $\sqrt{1-2x+x^3}-\sqrt[3]{1-3x+x^2}$,展开到含 x^5 的项。

命令如下:

```
x = sym('x');
f1 = (1 + x + x^2)/(1 - x + x^2);
f2 = sqrt(1 - 2 * x + x^3) - (1 - 3 * x + x^2)^(1/3);
taylor(f1,x,5)                                  %求(1)。展开到 x 的 4 次幂时应选择 n=5
ans =
1 + 2 * x + 2 * x^2 - 2 * x^4
taylor(f2,6)                                    %求(2)。
ans =
1/6 * x^2 + x^3 + 119/72 * x^4 + 239/72 * x^5
```

例 6.24 将下列多项式表示成 $x+1$ 的幂的多项式。

$$P(x)=1+3x+5x^2-2x^3$$

本例即要求在 $x=-1$ 处将函数 $P(x)$ 展开到 $n=4$。命令如下:

```
x = sym('x');
p = 1 + 3 * x + 5 * x^2 - 2 * x^3;
f = taylor(p,x,-1,4)
f =
-8 - 13 * x + 11 * (1 + x)^2 - 2 * (1 + x)^3
```

例 6.25 应用泰勒公式近似计算 $\sqrt[12]{4\,000}$。

因为 $4\,000=2^{12}-96=2^{12}(1-96\times2^{-12})$

所以

$$\sqrt[12]{4\ 000} = 2\sqrt[12]{1-96\times 2^{-12}}$$

命令如下：

```
x = sym('x');
f = (1 - x)^(1/12);                              %定义函数,4000^(1/12) = 2f(96/2^12)
g = taylor(f,4)                                  %求 f 的泰勒展开式 g,有 4000^(1/12) ≈ 2g(96/2^12)
g =
1 - 1/12 * x - 11/288 * x^2 - 253/10368 * x^3
b = 96/2^12;
a = 1 - b/12 - 11/288 * b^2 - 253/10368 * b^3    %计算 g(b)
a =
    0.9980
2 * a                                            %求 4000^(1/12)的结果
ans =
    1.9961
4000^(1/12)                                      %用 MATLAB 的乘方运算直接计算
ans =
    1.9961
```

用泰勒公式近似计算的结果和直接调用 MATLAB 函数的计算结果完全一致，说明这种计算方法是正确的。

读者也许注意到了，本例中在求得了函数 $f(x)$ 的泰勒展开式 $g(x)$ 后，没有直接使用 $g(b)$ 的语句计算函数 $g(x)$ 在 $x=b$ 处的值，而是将函数 g 重新输入了一次，这样做是很麻烦的，但没有捷径。符号函数不能像函数文件那样直接计算函数在指定点的值。

6.4.3 函数的傅立叶级数

设函数 $f(x)$ 和级数

$$s(x) = \frac{a_0}{2} + \sum_{k=1}^{+\infty}[a_k\cos(kx) + b_k\sin(kx)]$$

式中

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos kx\,\mathrm{d}x$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin kx\,\mathrm{d}x$$

如果级数 $s(x)$ 收敛于 $f(x)$，称 $s(x)$ 为 $f(x)$ 的傅立叶级数。从高等数学中知道，一般函数的傅立叶级数都是收敛的。傅立叶级数在电路分析、自动控制理论及通信分析中有广泛的应用。但 MATLAB 5.x 版中，尚未提供求函数傅立叶级数的内部函数。下面就设计一个简化的求任意函数的傅立叶级数的函数文件。

```
function mfourier = mfourier(f,n)
syms x a b c;
mfourier = int(f, - pi,pi)/2;                   %计算 a0
```

```
for i = 1:n
    a(i) = int(f * cos(i * x), - pi,pi);
    b(i) = int(f * sin(i * x), - pi,pi);
    mfourier = mfourier + a(i) * cos(i * x) + b(i) * sin(i * x);
end
return
```

调用该函数时,需给出被展开的符号函数 f 和展开项数 n,不可缺省。

例 6.26　在[$-\pi,\pi$]区间展开下列函数为傅立叶级数。

(1) $f(x) = x$,展开到第 5 项。

(2) $f(x) = |x|$,展开到第 5 项。

(3) $f(x) = \cos(ax)$,展开到第 6 项。

(4) $f(x) = \sin(bx)$,展开到第 4 项。

命令如下:

```
x = sym('x');a = sym('a');
f = x;
mfourier(f,5)                                  %求 f(x) = x 的傅立叶级数的前 5 项
ans =
2 * pi * sin(x) - pi * sin(2 * x) + 2/3 * pi * sin(3 * x) - 1/2 * pi * sin(4 * x) + 2/5 * pi * sin(5 * x)
f = abs(x);
mfourier(f,5)                                  %求 f(x) = |x|的傅立叶级数的前 5 项
ans =
1/2 * pi^2 - 4 * cos(x) - 4/9 * cos(3 * x) - 4/25 * cos(5 * x)
syms a;
f = cos(a * x);
mfourier(f,6)                                  %求 f(x) = cos(ax)的傅立叶级数的前 6 项
ans =
sin(pi * a)/a - 2/(a - 1)/(a + 1) * a * sin(pi * a) * cos(x) + 2/(a - 2)/(a + 2) * a * sin(pi * a) * cos(2 * x) -
2/(a - 3)/(a + 3) * a * sin(pi * a) * cos(3 * x) + 2/(a - 4)/(a + 4) * a * sin(pi * a) * cos(4 * x) - 2/(a - 5)/
(a + 5) * a * sin(pi * a) * cos(5 * x) + 2/(a - 6)/(a + 6) * sin(pi * a) * a * cos(6 * x)
f = sin(a * x);
mfourier(f,4)                                  %求 f(x) = sin(ax)的傅立叶级数的前 4 项
ans =
- 2/(a + 1)/(a - 1) * sin(pi * a) * sin(x) + 4/(a + 2)/(a - 2) * sin(pi * a) * sin(2 * x) - 6/(a + 3)/(a - 3) *
sin(pi * a) * sin(3 * x) + 8/(a + 4)/(a - 4) * sin(pi * a) * sin(4 * x)
```

上面定义的函数文件 mfourier.m 存在两个问题:第一调用参数规定太死,不能缺省;其次,只能在[$-\pi,\pi$]展开函数 $f(x)$,请读者对这个函数文件 mfourier.m 进行修改,使其能在一般的[$-1,1$]上将 $f(x)$展开为傅立叶级数,并且调用参数更灵活、更方便。

6.5 代数方程的符号求解

所谓代数方程是指未涉及微积分运算的方程。与微分方程相比,代数方程比较简单,但除一

些特殊形式的代数方程外,要求得一个代数方程的解也并非易事。在第 5 章,已经讨论了代数方程的数值求解方法。本节讨论在 MATLAB 中,怎样求解用符号表达式表示的一般代数方程。

6.5.1 线性方程组的符号求解

第 5 章已经讨论过,对于形如 $\boldsymbol{AX}=\boldsymbol{b}$ 的线性方程组,有

X = inv(A) * b 或 X = A \ b

对于数值计算,$\boldsymbol{A}$,$\boldsymbol{b}$ 必须是数值矩阵,不能含有符号量。在下面的讨论中去掉这一限制,允许矩阵 $\boldsymbol{A}$,$\boldsymbol{b}$ 中含有符号量。

MATLAB 中提供了一个求解线性代数方程组的函数 linsolve,其调用格式为:

linsolve(A,b)

该函数返回线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的解。这里 $\boldsymbol{A}$,$\boldsymbol{b}$ 允许是符号矩阵,$\boldsymbol{A}$ 必须行满秩(即 rank(A) >= m,m 是 $\boldsymbol{A}$ 的行数)。

例 6.27 求线性方程组 $\boldsymbol{AX}=\boldsymbol{b}$ 的解。

(1) $\boldsymbol{A}=\begin{bmatrix}34 & 8 & 4\\3 & 34 & 3\\3 & 6 & 8\end{bmatrix}$, $\boldsymbol{b}=\begin{bmatrix}4\\6\\2\end{bmatrix}$

(2) $\boldsymbol{A}=\begin{bmatrix}a11 & a12 & a13\\a21 & a22 & a23\\a31 & a32 & a33\end{bmatrix}$, $b=\begin{bmatrix}b1\\b2\\b3\end{bmatrix}$

解方程组(1)的命令如下:

```
A = [34,8,4;3,34,3;3,6,8];
b = [4;6;2];
X = linsolve(A,b)                          %调用 linsolve 函数求(1)的解
X =
[138/2045]
[  66/409]
[ 212/2045]
A \ b                                      %用另一种方法求(1)的解
ans =
    0.0675
    0.1614
    0.1037
```

上述两种结果本质上相同,但形式不同,因为符号运算将数值结果都用精确值表示,不化为小数,而数值运算时都有用限小数表示结果。

解方程组(2)的命令如下:

```
syms a11 a12 a13 a21 a22 a23 a31 a32 a33 b1 b2 b3;
A = [a11,a12,a13;a21,a22,a23;a31,a32,a33];
b = [b1;b2;b3];
X = linsolve(A,b)                          %调用 linsolve 函数求(2)的解
XX = A \ b                                 %用左除运算求(2)的解
```

读者可以上机运行上述命令，仔细比较会发现，两种方法得到的结果是完全相同的。

6.5.2 非线性方程组的符号求解

第5章讨论过非线性方程组的数值求解方法。非线性方程组的一般形式是：

$$\boldsymbol{F}(\boldsymbol{X})=\boldsymbol{0}$$

其中 $\boldsymbol{X}$ 是未知量，是一个含 n 个元素的列向量，$\boldsymbol{0}$ 是含 n 个元素的 $\boldsymbol{0}$ 向量。在5.4节中，要求向量值函数 $\boldsymbol{F}(\boldsymbol{X})$ 中不含符号量，即 $\boldsymbol{F}$ 函数中不含参数，这在实际工作中是难以做到的。下面的讨论中，假设 $\boldsymbol{F}$ 是一个任意的向量值函数。

求解非线性方程组的函数是 solve，调用格式为：

```
solve('eqn1','eqn2',…,'eqnN','var1,var2,…,varN')
```

该函数求解由参数 *eqn*1、*eqn*2、…、*eqnN* 所定义的方程组，求解变量由 *var*1、*var*2、…、*varN* 等定义，若不指定求解变量，由缺省规则确定。

当方程右端为0时，方程 *eqni* 中可以不包含右端项和等号，而仅列出方程左端的表达式。

例 6.28 解下列方程。

(1) $\dfrac{1}{x+2}+\dfrac{4x}{x^2-4}=1+\dfrac{2}{x-2}$

(2) $x-\sqrt[3]{x^3-4x-7}=1$

(3) $2\sin(3x-\pi/4)=1$

(4) $x+xe^x-10=0$

命令如下：

```
x = solve('1/(x+2)+4*x/(x^2-4)=1+2/(x-2)','x')        %解方程(1)
x =
1
f = sym('x-(x^3-4*x-7)^(1/3)=1');
x = solve(f)                                          %解方程(2)
x =
3
x = solve('2*sin(3*x-pi/4)=1')                        %解方程(3)
x =
5/36*pi
x = solve('x+x*exp(x)-10','x')                        %解方程(4)。仅标出方程的左端
x =
1.6335061701558463841931651789789
```

用函数 solve 求解方程时，常常会发生遗根的情况，分析一下上面的输出结果，就会发现例题解答中给出的根不完整，有遗漏。对于无任何特征的一般方程组，要保证逐一求找出其全部根，是一个非常困难的问题，人们期待后续的 MATLAB 版本会弥补现在留给用户的这一缺陷。

例 6.29 求下列方程组的解。

(1) $\begin{cases}\dfrac{1}{x^3}+\dfrac{1}{y^3}=28\\[2mm]\dfrac{1}{x}+\dfrac{1}{y}=4\end{cases}$ (2) $\begin{cases}x+y=98\\\sqrt[3]{x}+\sqrt[3]{y}=2\end{cases}$

(3) $\begin{cases} u^3 + v^3 = 98 \\ u + v = 2 \end{cases}$　　(4) $\begin{cases} x^2 + y^2 = 5 \\ 2x^2 - 3xy - 2y^2 = 0 \end{cases}$

命令如下:

```
[x y] = solve('1/x^3 + 1/y^3 = 28','1/x + 1/y = 4','x,y')          %解方程组(1)
x =
[  1]
[1/3]
y =
[1/3]
[  1]
[x y] = solve('x + y - 98','x^(1/3) + y^(1/3) - 2','x,y')          %解方程组(2)
Warning: Explicit solution could not be found.
> In C:\MATLABR11\toolbox\symbolic\solve.m at line 136
x =
[empty sym]
y =
  []
```

对方程组(2)MATLAB 给出了无解的结论,显然错误,下面看完全与其同构的方程组(3)。输入命令如下:

```
[u,v] = solve('u^3 + v^3 - 98','u + v - 2','u,v')                  %解方程组(3)
u =
[  5]
[ -3]
v =
[ -3]
[  5]
[x v] = solve('x^2 + y^2 - 5','2 * x^2 - 3 * x * y - 2 * y^2')      %解方程组(4)
x =
[ -1]
[  1]
[  2]
[ -2]
v =
[  2]
[ -2]
[  1]
[ -1]
```

通过上面的例子,读者应该对 MATLAB 求解方程组的强大能力有所了解,同时对其不足也有认识。在以后遇到方程组求解问题时,应充分发挥 MATLAB 的功能,但当由 MATLAB 给出无解或未找到所期望的解时,不要认为原方程组就真正无解了,还需要用其他方法试探着求解。如果知道方程组在某点附近有解,不妨用方程组的数值求解函数 fsolve 试探求解,一般能找到所期望的

解。总之,方程组求解是一个古老而又困难的问题,MATLAB 给人们提供了一种虽然不是万能但往往非常有效的求解手段。

6.6 常微分方程的符号求解

第 5 章讨论了常微分方程的数值求解函数 ode23 和 ode45。数值求解只能给出方程的数值解,与读者在微分方程课程中所学习的求解方法和解的结果还不相同。MATLAB 能否对常微分方程提供连续的符号解呢? 这就是本节要讨论的问题。MATLAB 的符号运算工具箱中提供了功能强大的求解常微分方程的函数 dsolve,在介绍 dsolve 的用法之前,先简要说明在函数 dsolve 中怎样表示微分运算。

MATLAB 中表示微分方程时,用大写的字母 D 表示导数。例如,Dy 表示 y',D2y 表示 y''。Dy(0) = 5表示 $y'(0) = 5$。D3y + D2y + Dy - x + 5 = 0 表示微分方程 $y''' + y'' + y' - x + 5 = 0$。

有了上面的介绍后,就可以介绍 dsolve 函数的用法了。该函数的调用格式为:

```
dsolve('eqn1','condition','var')
```

该函数求解微分方程 eqn1 在初值条件 condition 下的特解。参数 var 描述方程中的自变量符号,省略时按缺省原则处理,若没有给出初值条件 condition,则求方程的通解。

dsolve 在求解微分方程组时的调用格式为:

```
dsolve('eqn1','eqn2',…,'eqnN','condition1',…,'conditionN','var1',…,'varN')
```

函数求解微分方程组 *eqn*1、…、*eqnN* 在初值条件 *conditoion*1、…、*conditionN* 下的解,若不给出初值条件,则求方程组的通解。*var*1、…、*varN* 给出求解变量。

6.6.1 求常微分方程的通解

在调用 dsolve 函数时,若不写 condition 参数就可求常微分方程的通解。

例 6.30 求下列微分方程的通解。

(1) $\frac{dy}{dx} = \frac{x^2 + y^2}{2x^2}$

(2) $x^2 \frac{dy}{dx} + 2xy = e^x$

(3) $\frac{dy}{dx} = \frac{x}{y\sqrt{1 - x^2}}$

命令如下:

```
y = dsolve('Dy - (x^2 + y^2)/x^2/2','x')          %解(1)。方程的右端为 0 时可以不写
y =
x * ( - log(x) + 2 + C1)/( - log(x) + C1)
y = dsolve('Dy * x^2 + 2 * x * y - exp(x)','x')    %解(2)
y =
1/x^2 * exp(x) + 1/x^2 * C1
y = dsolve('Dy - x/y/sqrt(1 - x^2)','x')          %解(3)
```

```
y =
[ ( - 2 * ( - (x - 1) * (x + 1))^(1/2) + C1)^(1/2)]
[ - ( - 2 * ( - (x - 1) * (x + 1))^(1/2) + C1)^(1/2)]
```

6.6.2　求常微分方程的特解

求解常微分方程特解时,只要将初值条件代入函数 dsolve 的 condition 参数中即可。

例 6.31　求微分方程的特解。

(1) $\frac{dy}{dx}=2xy^2,\quad y(0)=1$

(2) $\frac{dy}{dx}=\frac{x^2}{1+y^2},\quad y(2)=1$

命令如下:

```
y = dsolve(' Dy = 2 * x * y^2 ',' y(0) = 1 ',' x ')        %解(1)
y =
 - 1/(x^2 - 1)
y = dsolve(' Dy = x^2/(1 + y^2)',' y(2) = 1 ',' x ')      %解(2)
y =
[ 1/2 * (4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))^(1/3) - 2/(4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))^
(1/3)]
[ - 1/4 * (4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))^(1/3) + 1/(4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))
^(1/3) + i * ( - 1/4 * (4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))^(1/3) - 1/(4 * x^3 - 16 + 4 * (20 + x^6 - 8 *
x^3)^(1/2))^(1/3)) * 3^(1/2)]
[ - 1/4 * (4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))^(1/3) + 1/(4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))
^(1/3) + i * (1/4 * (4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^3)^(1/2))^(1/3) + 1/(4 * x^3 - 16 + 4 * (20 + x^6 - 8 * x^
3)^(1/2))^(1/3)) * 3^(1/2)]
```

这里给出的解很冗长。如果读者熟悉微分方程的解法,可以很快求得方程(2)的用隐式函数表示的解

$$y^3-x^3+3y+4=0$$

求解这个三次方程,解出 y,得到与 MATLAB 一致的 3 个解。

例 6.32　用微分方程的数值解法和符号解法解下列方程,并对结果进行比较。

$$x\frac{dy}{dx}+2y=4x^2, y(1)=2$$

在 MATLAB 命令窗口,输入命令:

```
y = dsolve(' Dy + 2 * y/x - 4 * x ',' y(1) = 2 ',' x ')        %用符号方法得到方程的解析解
y =
x^2 + 1/x^2
```

为了求方程的数值解,需要按要求建立一个函数文件 fxyy.m:

```
function f = fxyy(x,y)
f = (4 * x^2 - 2 * y)/x;      %只能是 y ' = f(x,y)的形式,当不是这种形式时,要变形。
return
```

输入命令:

```
[t,w] = ode45('fxyy',[1,2],2);        % 得到区间[1,2]中的数值解,以向量 t、w 存储。
```

为了对两种结果进行比较,在同一个坐标系中作出两种结果的图形。输入命令:

```
x = linspace(1,2,100);
y = x.^2 + 1./x.^2;                    % 为作图把符号解的结果离散化
plot(x,y,'b.',t,w,'r-');
```

符号解图形用兰色圆点表示,数值解图形用红色实线表示。结果如图 6.1 所示。可以看出,两种结果的图形是完全一致的,这也说明了数值解法有足够的精确度。

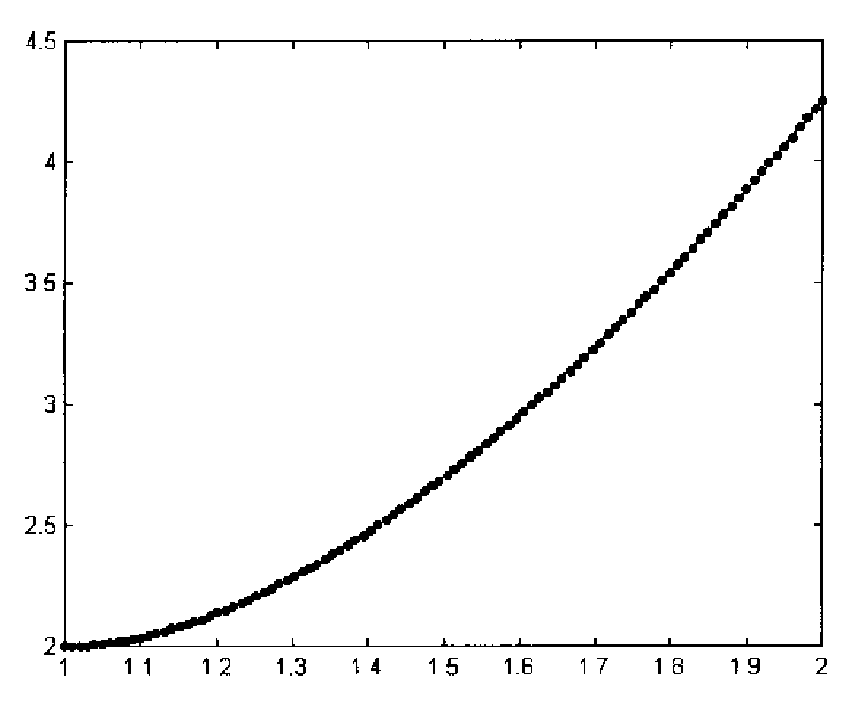


图 6.1 数值解和符号解比较

6.6.3 常微分方程组求解

在一个问题中如有几个未知函数,因而需要由几个微分方程进行约束,这样的方程组称为微分方程组。例如

$$\begin{cases} \dfrac{\mathrm{d}f}{\mathrm{d}x} = g(x) - f(x) + x^2 \\ \dfrac{\mathrm{d}g}{\mathrm{d}x} = -2g(x) - x \end{cases}$$

有两个未知函数 $f(x)$、$g(x)$,由两个方程共同约束,这两个方程组成一个微分方程组。dsolve 函数的第 2 种格式就是专门用来求解微分方程组的。

例 6.33 求下列微分方程组的解。

(1) $\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = 4x - 2y \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = 2x - y \end{cases}$　　(2) $\begin{cases} \dfrac{\mathrm{d}^2x}{\mathrm{d}t^2} - y = 0 \\ \dfrac{\mathrm{d}^2y}{\mathrm{d}t^2} + x = 0 \end{cases}$

命令如下:

```
[x,y] = dsolve('Dx = 4 * x - 2 * y','Dy = 2 * x - y','t')   %解方程组(1)
x =
- 1/3 * C1 + 4/3 * C1 * exp(3 * t) - 2/3 * C2 * exp(3 * t) + 2/3 * C2
y =
2/3 * C1 * exp(3 * t) - 2/3 * C1 + 4/3 * C2 - 1/3 * C2 * exp(3 * t)
[x,y] = dsolve('D2x - y','D2y + x','t')                      %解方程组(2)
```

因为第(2)题的结果非常冗长,所以书中未予列出。许多情况下若用隐函数表示微分方程的结果会比显式方法简洁,但在 MATLAB 中,dsolve 函数一律使用显式方式表示结果,这也是符号方法求解微分方程的一个特点。

符号运算功能强大,是 MATLAB 的一个特色。对于许多工程计算问题,MATLAB 中既提供了数值计算方法,也提供了符号计算方法,这两者各有用途:数值计算方法的主要特征是"可行性",但不能给出解析解;符号方法能给出解析解,但结果一般较复杂、冗长,且对问题的选择性强,很多问题(特别是求解微分方程的问题)无法用符号方法求解。读者应根据问题灵活选择解决方案。一般情况下,符号方法使用要方便一些。

习 题 六

1. 分解因式。

(1) x^9-1　　(2) $x^4+x^3+2x^2+x+1$

(3) $125x^6+75x^4+15x^2+1$　　(4) $x^2+y^2+z^2+2(xy+yz+zx)$

2. 化简表达式。

(1) $\frac{y}{x}+\frac{x}{y}$　　(2) $\sqrt{\frac{a+\sqrt{a^2-b}}{2}}+\sqrt{\frac{a-\sqrt{a^2-b}}{2}}$

(3) $2\cos^2x-\sin^2x$　　(4) $\sqrt{3+2\sqrt{2}}$

3. 求函数的极限。

(1) $\lim\limits_{x\to4}\frac{x^2-6x+8}{x^2-5x+4}$　　(2) $\lim\limits_{x\to0^-}\frac{|x|}{x}$

(3) $\lim\limits_{x\to0}\frac{\sqrt{1+x^2}-1}{x}$　　(4) $\lim\limits_{x\to\infty}\left(x+\frac{1}{x}\right)^x$

4. 求函数的符号导数。

(1) $y=3x^2-5x+1$,求 y'、y''。　　(2) $y=\sqrt{x+\sqrt{x+\sqrt{x}}}$,求 y'、y''。

(3) $y=\sin x-\frac{x^2}{2}$,求 y'、y''。　　(4) $z=x+y-\sqrt{x^2+y^2}$,求$\frac{\partial z}{\partial x}$、$\frac{\partial z}{\partial y}$。

5. 求不定积分。

(1) $\int\frac{dx}{x+a}$　　(2) $\int\sqrt[3]{1-3x}dx$

(3) $\int\frac{dx}{\sin^2x\cos^2x}$　　(4) $\int\frac{x^2dx}{\sqrt{a^2+x^2}}$

6. 用数值与符号两种方法求给定函数的定积分,并对结果进行比较。

(1) $\int_0^1 x(2-x^2)^{12}dx$　　(2) $\int_{-1}^1\frac{x\,dx}{x^2+x+1}$

(3) $\int_0^{\pi}(x\sin x)^2 \mathrm{d}x$

(4) $\int_{\frac{1}{e}}^{e} |\ln x| \mathrm{d}x$

7. 求下列级数之和。

(1) $1-\frac{3}{2}+\frac{5}{4}-\frac{7}{8}+\cdots$

(2) $x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\cdots$

(3) $1+\frac{1}{9}+\frac{1}{25}+\frac{1}{49}+\cdots$

(4) $\frac{1}{1\times2\times3}+\frac{1}{2\times3\times4}+\frac{1}{3\times4\times5}+\frac{1}{4\times5\times6}+\cdots$

8. 求函数在 $x=x_0$ 的泰勒展开式。

(1) $y=x^4-5x^3+x^2-3x+4, x_0=4$

(2) $y=\frac{e^x+e^x}{2}, x_0=0, n=5$

(3) $y=\tan x, x_0=2, n=3$

(4) $y=\sin^2 x, x_0=0, n=8$

(5) $y=\sqrt{x^3+x^2+5x+3}, x_0=0, n=5$

9. 求函数的傅立叶展开式。

(1) $y=2x^3(-\pi\leqslant x\leqslant\pi), n=5$

(2) $y=e^{-x}+x(-\pi\leqslant x\leqslant\pi), n=5$

10. 求微分方程初值问题的符号解，并与数值解进行比较。

$xy''+(1-n)y'+y=0$

$y(0)=y'(0)=0$

11. 求非线性方程的符号解。

(1) $ax^2+bx+c=0$

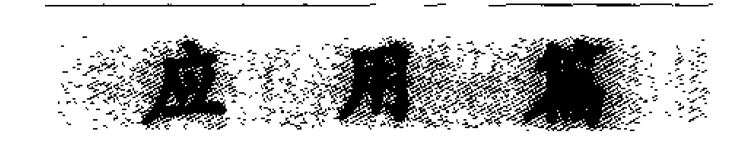
(2) $2\sin\left(3x+\frac{\pi}{4}\right)=1$

(3) $\sin x-\sqrt{3}\cos x=\sqrt{2}$

(4) $x^2+10(x-1)\sqrt{x}+14x+1=0$

12. 求非线性方程组的符号解。

$$\begin{cases}\frac{4x^2}{4x^2+1}=y\\ \frac{4y^2}{4y^2+1}=z\\ \frac{4z^2}{4z^2+1}=x\end{cases}$$


应用篇

第7章　MATLAB图形用户界面设计

所谓图形用户界面(Graphical User Interface,简称GUI)是指由窗口、菜单、对话框等各种图形元素组成的用户界面。在这种用户界面下,用户的操作既形象生动,又方便灵活,所以在人机交互方式中占据主导地位。诚然,如果所解决的问题输入输出比较单一,那么一般不会考虑图形用户界面的设计,但如果要开发一个通用软件,那么采用图形用户界面是人机交互方式的最佳选择。

MATLAB图形用户界面的设计是通过图形窗口、用户菜单、用户控件等图形对象的操作来实现的,在第4章已经介绍了图形窗口、坐标轴等图形对象的操作,这些内容是本章学习的基础,务必仔细阅读并掌握。本章重点介绍如何通过用户菜单对象来建立自己的菜单系统、如何通过用户控件对象来建立对话框,最后介绍MATLAB提供的用户界面设计工具。

7.1　菜单设计

菜单是一种十分重要的用户界面,也是软件的一种重要操作方式,现代软件大都具有十分友好的菜单系统。从图4.26图形对象的树形结构可知,MATLAB用户菜单对象是图形窗口的子对象,所以菜单设计总在某一个图形窗口中进行。MATLAB的各个图形窗口有自己的菜单栏,包括File、Edit、Tools、Windows和Help共5个菜单项。为了建立用户自己的菜单系统,可以先将图形窗口的MenuBar属性设置为none,以取消图形窗口缺省的菜单,然后再建立用户自己的菜单。

7.1.1　用户菜单的建立

用户菜单通常包括一级菜单(菜单条)和二级菜单,有时根据需要还可以往下建立子菜单(三级菜单等),每一级菜单又包括若干菜单项。要建立用户菜单可用uimenu函数,因其调用方法不同,该函数可以用于建立一级菜单项和子菜单项。

建立一级菜单项的函数调用形式为:

一级菜单项句柄 = uimenu(图形窗口句柄,属性名1,属性值1,属性名2,属性值2,…)

建立子菜单项的函数调用形式为:

子菜单项句柄 = uimenu(一级菜单项句柄,属性名1,属性值1,属性名2,属性值2,…)

这两种调用形式的区别在于:建立一级菜单项时,要给出图形窗口的句柄值。如果省略了这个句柄值,MATLAB就在当前图形窗口中建立这个菜单项。如果此时不存在活动图形窗口,MATLAB会自动打开一个图形窗口,并将该菜单项作为它的菜单对象。在建立子菜单项时,必须指定一级菜单项对应的句柄值。例如

```
hm = uimenu(gcf,'Label','File');
hm1 = uimenu(hm,'Label','Save');
```

```
hm2 = uimenu(hm,' Label ',' Save As ');
```

将在当前图形窗口菜单条中建立名为 File 的菜单项。其中,Label 属性值 File 就是菜单项的名字,hm 是 File 菜单项的句柄值,供定义该菜单项的子菜单之用。后两条命令将在 File 菜单项下建立 Save 和 Save As 两个子菜单项。

7.1.2 菜单对象常用属性

菜单对象具有 Children、Parent、Tag、Type、UserData、Visible 等公共属性,它们的含义请阅读 4.5 节有关内容。除公共属性外,还有一些常用的特殊属性。

(1) Label 属性。该属性的取值是字符串,用于定义菜单项的名字。可以在字符串中加入 & 字符,这时在该菜单项名字上,跟随 & 字符后的字符有一条下划线,& 字符本身不出现在菜单项中。对于这种有带下划线字符的菜单,可以用 Alt 键加该字符键来激活相应的菜单项。

(2) Accelerator 属性。该属性的取值可以是任何字母,用于定义菜单项的快捷键。如取字母 W,则表示定义快捷键为 Ctrl + W。

(3) Callback 属性。该属性的取值是字符串,可以是某个 M 文件名或一组 MATLAB 命令。在该菜单项被选中以后,MATLAB 将自动地调用此回调函数来作出对相应菜单项的响应,如果没有设置一个合适的回调函数,则此菜单项也将失去其应有的意义。

在产生子菜单时 Callback 选项也可以省略,因为这时可以直接打开下一级菜单,而不是侧重于对某一函数进行响应。

(4) Checked 属性。该属性的取值是 on 或 off(缺省值),该属性为菜单项定义一个指示标记,可以用这个特性指明菜单项是否已选中。

(5) Enable 属性。该属性的取值是 on(缺省值)或 off,这个属性控制菜单项的可选择性。如果它的值是 off,则此时不能使用该菜单。此时,该菜单项呈灰色。

(6) Position 属性。该属性的取值是数值,它定义一级菜单项在菜单条上的相对位置或子菜单项在菜单组内的相对位置。例如,对于一级菜单项,若 Position 属性值为 1,则表示该菜单项位于图形窗口菜单条的可用位置的最左端。

(7) Separator 属性。该属性的取值是 on 或 off(缺省值)。如果该属性值为 on,则在该菜单项上方添加一条分隔线,可以用分隔线将各菜单项按功能分开。

例 7.1 建立图 7.1 所示的"图形演示系统"菜单。菜单条中含有 3 个菜单项:Plot、Option 和 Quit。Plot 中有 Sine Wave 和 Cosine Wave 两个子菜单项,分别控制在本图形窗口画出正弦和余弦曲线。Option 菜单项的内容如图 7.1 所示。其中 Grid on 和 Grid off 控制给坐标轴加网格线,Box on 和 Box off 控制给坐标轴加边框,而且这 4 项只有在画有曲线时才是可选的。Figure Color 控制图形窗口背景颜色。Quit 控制是否退出系统。

程序如下:

```
screen = get(0,' ScreenSize ');
W = screen(3);H = screen(4);
figure(' Color ',[1,1,1],' Position ',[0.2 * H,0.2 * H,0.6 * W,0.4 * H],...
      ' Name ','图形演示系统',' NumberTitle ',' off ',' MenuBar ',' none ');
%定义 Plot 菜单项
hplot = uimenu(gcf,' Label ',' &Plot ');
```

```
uimenu(hplot,'Label','Sine Wave','Call',['t = -pi:pi/20:pi;','plot(t,sin(t));',...
    'set(hgon,"Enable","on");','set(hgoff,"Enable","on");',...
    'set(hbon,"Enable","on");','set(hboff,"Enable","on");']);
uimenu(hplot,'Label','Cosine Wave','Call',['t = -pi:pi/20:pi;','plot(t,cos(t));',...
    'set(hgon,"Enable","on");','set(hgoff,"Enable","on");',...
    'set(hbon,"Enable","on");','set(hboff,"Enable","on");']);
%定义 Option 菜单项
hoption = uimenu(gcf,'Label','&Option');
hgon = uimenu(hoption,'Label','&Grig on','Call','grid on','Enable','off');
hgoff = uimenu(hoption,'Label','&Grig off','Call','grid off','Enable','off');
hbon = uimenu(hoption,'Label','&Box on','separator','on','Call','box on','Enable','off');
hboff = uimenu(hoption,'Label','&Box off','Call','box off','Enable','off');
hfigcor = uimenu(hoption,'Label','&Figure Color','Separator','on');
uimenu(hfigcor,'Label','&Red','Accelerator','r','Call','set(gcf,"Color","r");');
uimenu(hfigcor,'Label','&Blue','Accelerator','b','Call','set(gcf,"Color","b");');
uimenu(hfigcor,'Label','&Yellow','Call','set(gcf,"Color","y");');
uimenu(hfigcor,'Label','&White','Call','set(gcf,"Color","w");');
%定义 Quit 菜单项
uimenu(gcf,'Label','&Quit','Call','close(gcf)');
```

程序运行后可以建立图7.1所示的菜单。

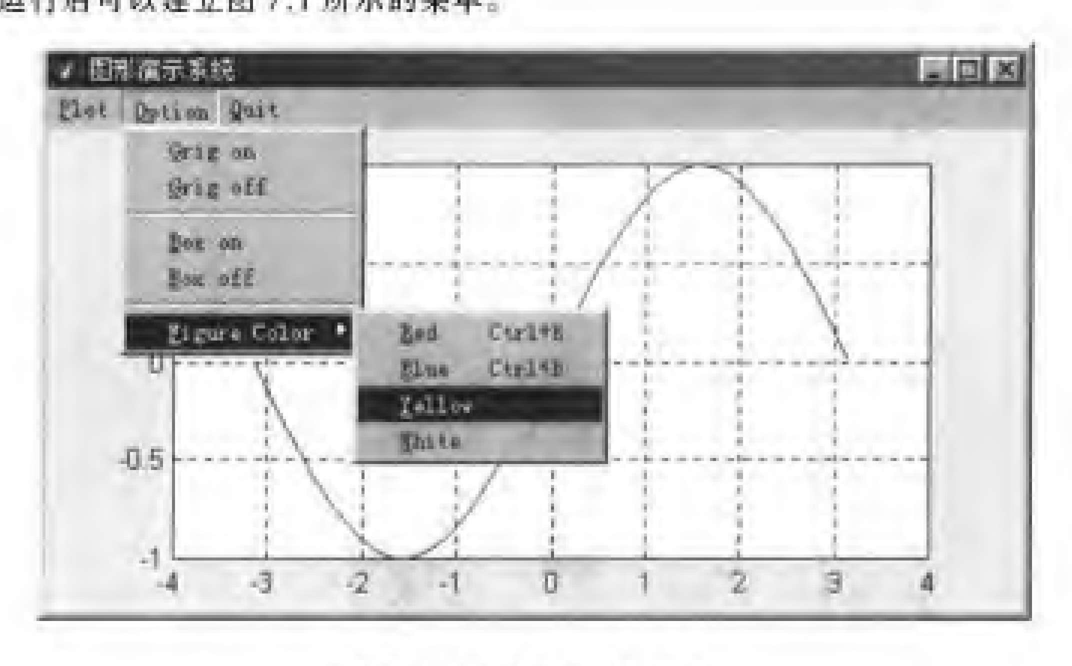


图7.1 "图形演示系统"菜单

7.1.3 快捷菜单

快捷菜单是用鼠标右键单击某对象时在屏幕上弹出的菜单。这种菜单出现的位置是不固定的，而且总是和某个图形对象相联系。在MATLAB中，可以使用uicontextmenu函数和图形对象的UIContextMenu属性来建立快捷菜单，具体步骤为：

(1) 利用 uicontextmenu 函数建立快捷菜单。

(2) 利用 uimenu 函数为快捷菜单建立菜单项。

(3) 利用 set 函数将该快捷菜单和某图形对象联系起来。

例 7.2 绘制曲线 $y=2e^{-0.5x}\sin(2\pi x)$,并建立一个与之相联系的快捷菜单,用以控制曲线的线型和曲线宽度。

程序如下:

```
x = 0:pi/100:2 * pi;
y = 2 * exp( - 0.5 * x). * sin(2 * pi * x);
hl = plot(x,y);
hc = uicontextmenu;                          %建立快捷菜单
hls = uimenu(hc,'Label','线型');             %建立菜单项
hlw = uimenu(hc,'Label','线宽');
uimenu(hls,'Label','虚线','Call','set(hl,''LineStyle'','':'');');
uimenu(hls,'Label','实线','Call','set(hl,''LineStyle'',''-'');');
uimenu(hlw,'Label','加宽','Call','set(hl,''LineWidth'',2);');
uimenu(hlw,'Label','变细','Call','set(hl,''LineWidth'',0.5);');
set(hl,'UIContextMenu',hc);                  %将该快捷菜单和曲线对象联系起来
```

程序运行后先按缺省参数(0.5 磅实线)画线,若将鼠标指针指向线条并单击右键,则弹出快捷菜单(如图 7.2 所示),选择菜单命令可改变线型和曲线宽度。

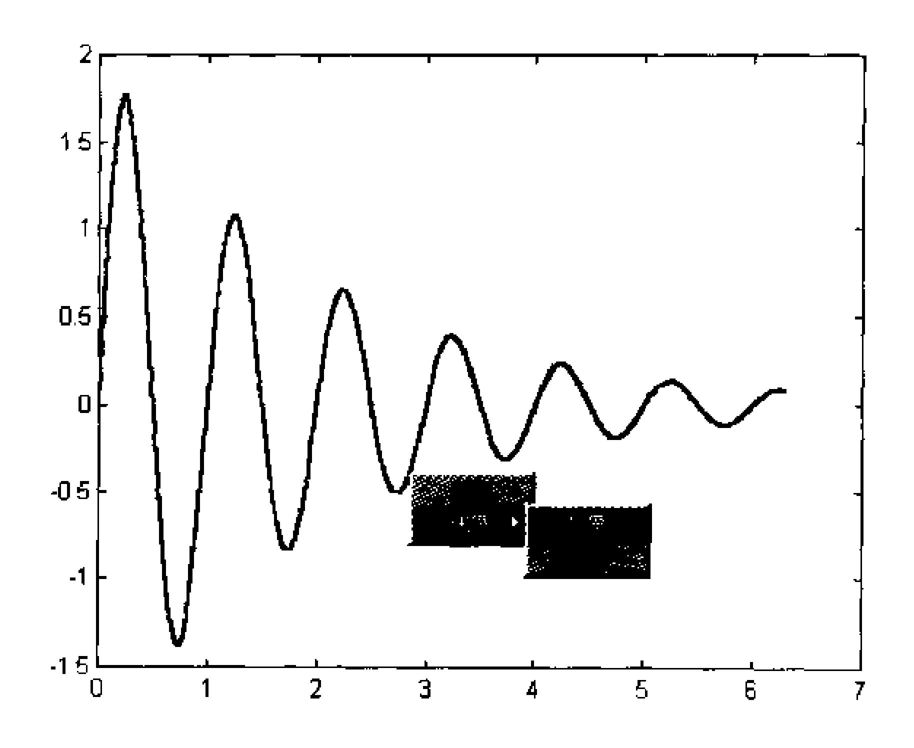


图 7.2 快捷菜单设计

7.2 对话框设计

对话框是用户与计算机进行信息交流的临时窗口，在现代软件中有着广泛的应用。在软件设计时，借助于对话框可以更好地满足用户操作需要，使用户操作更加方便灵活。

7.2.1 对话框的控件

在对话框上有各种各样的控件，利用这些控件可以实现有关控制。下面先介绍这些控件。

(1) 按钮(Push Button)。按钮是对话框中最常用的控件对象，其特征是在矩形框上加上文字说明。一个按钮代表一种操作，所以有时也称命令按钮。

(2) 双位按钮(Toggle Button)。在矩形框上加上文字说明。这种按钮有 2 个状态，即按下状态和弹起状态。每单击一次其状态将改变一次。

(3) 单选按钮(Radio Button)。单选按钮是一个圆圈加上文字说明。它是一种选择性按钮，当被选中时，圆圈的中心有一个实心的黑点，否则圆圈为空白。在一组单选按钮中，通常只能有一个被选中，如果选中了其中一个，则原来被选中的就不再处于被选中状态，这就像收音机一次只能选中一个电台一样，故称作单选按钮。在有些文献中，也称作无线电按钮或收音机按钮。

(4) 复选框(Check Box)。复选框是一个小方框加上文字说明。它的作用和单选按钮相似，也是一组选择项，被选中的项其小方框中有√。与单选按钮不同的是，复选框一次可以选择多项。这也是“复选框”名字的由来。有些文献中也称作检测框。

(5) 列表框(List Box)。列表框列出可供选择的一些选项，当选项很多而列表框装不下时，可使用列表框右端的滚动条进行选择。

(6) 弹出框(Popup Menu)。弹出框平时只显示当前选项，单击其右端的向下箭头即弹出一个列表框，列出全部选项。其作用与列表框类似。

(7) 编辑框(Edit Box)。编辑框可供用户输入数据用。在编辑框内可提供缺省的输入值，随后用户可以进行修改。

(8) 滑动条(Slider)。滑动条可以用图示的方式输入指定范围内的一个数量值。用户可以移动滑动条中间的游标来改变它对应的参数。

(9) 静态文本(Static Text)。静态文本是在对话框中显示的说明性文字，一般用来给用户作必要的提示。因用户不能在程序执行过程中改变文字说明，故将其称为静态文本。

(10) 边框(Frame)。边框主要用于修饰用户界面，使用户界面更友好。可以用边框在图形窗口中圈出一块区域，而将某些控件对象组织在这块区域中。

7.2.2 对话框的设计

在 MATLAB 中，要设计一个对话框，首先要建立一个图形窗口，然后在图形窗口中放置有关的用户控件对象。

1. 建立控件对象

MATLAB 提供了用于建立控件对象的函数 uicontrol，其调用格式为：

对象句柄 = uicontrol(图形窗口句柄,属性名 1,属性值 1,属性名 2,属性值 2,…)

其中各个属性名及可取的值和前面介绍的 uimenu 函数相似,但也不尽相同,下面将介绍一些常用的属性。

2. 控件对象的属性

MATLAB 的 10 种控件对象使用相同的属性类型,但是这些属性对于不同类型的控件对象,其含义不尽相同。除 Children、Parent、Tag、Type、UserData、Visible 等公共属性外,还有一些常用的特殊属性。

(1) Position 属性。该属性的取值是一个由 4 个元素构成的向量,其形式为[n1,n2,n3,n4]。这个向量定义了控件对象在屏幕上的位置和大小,其中 $n1$ 和 $n2$ 分别为控件对象左下角相对于图形窗口的横纵坐标值,$n3$ 和 $n4$ 分别为控件对象的宽度和高度。它们的单位由 Units 属性决定。

(2) Units 属性。该属性的取值可以是 pixel(像素,为缺省值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)或 points(磅)。除了 normalized 以外,其他单位都是绝对度量单位。所有单位的度量都是从图形窗口的左下角处开始,在相对单位下,图形窗口的左下角对应为(0,0),而右上角对应为(1.0,1.0)。该属性将影响一切定义大小的属性项,如前面的 Position 属性。

(3) Callback 属性。该属性的取值是字符串。和 uimenu 函数一样,Callback 属性允许用户建立起在对话框控件对象被选中后的响应命令。

(4) String 属性。该属性的取值是字符串。它定义控件对象的说明文字,如按钮上的说明文字以及单选按钮或复选按钮后面的说明文字等。

(5) Style 属性。该属性的取值可以是 push(按钮,缺省值)、toggle(双位按钮)、radio(单选按钮)、check(复选框)、list(列表框)、popup(弹出框)、edit(编辑框)、text(静态文本)、slider(滑动条)和 frame(边框)。这个属性定义控件对象的类型。

(6) BackgroundColor 属性。该属性的取值是代表某种颜色的字符或 RGB 三元组。它定义控件对象区域的背景色,它的缺省颜色是浅灰色。

(7) ForegroundColor 属性。该属性的取值与 BackgroundColor 属性相同。ForegroundColor 属性定义控件对象说明文字的颜色,其缺省颜色是黑色。

(8) Max、Min 属性。Max 和 Min 属性的取值都是数值,其缺省值分别是 1 和 0。这两个属性值对于不同的控件对象类型,其意义是不同的。以下分别予以介绍:

当单选按钮被激活时,它的 Value 属性值为 Max 属性定义的值。当单选按钮处于非激活状态时,它的 Value 属性值为 Min 属性定义的值。

当复选框被激活时,它的 Value 属性值为 Max 属性定义的值。当复选框处于非激活状态时,它的 Value 属性值为 Min 属性定义的值。

对于滑动条对象,Max 属性值必须比 Min 属性值大,Max 定义滑动条的最大值,Min 定义滑动条的最小值。

对于编辑框,如果 $Max - Min > 1$,那么对应的编辑框接受多行字符输入。如果 $Max - Min \leq 1$,那么编辑框仅接收单行字符输入。

对于列表框,如果 $Max - Min > 1$,那么在列表框中允许多项选择;如果 $Max - Min \leq 1$,那么在列表框中只允许单项选择。

另外,边框、弹出框和静态文本等控件对象不使用Max和Min属性。

(9) Value属性。该属性的取值可以是向量值,也可以是数值。它的含义依赖于控件对象的类型。对于单选按钮和复选框,当它们处于激活状态时,Value属性值由Max属性值定义,反之由Min属性定义。对于弹出框,Value属性值是被选项的序号,所以由Value的值,可知弹出框的选项。同样,对于列表框,Value属性值定义了列表框中高亮度选项的序号。对于滑动条对象,Value属性值处于Min与Max属性值之间,由滑动条标尺位置对应的值定义。其他的控件对象不使用这个属性值。

(10) FontAngle属性。该属性的取值是normalized(缺省值)、italic和oblique。这个属性值定义控件对象标题等的字体。其值为normalized时,选用系统缺省的正字体,而其值为italic或oblique时,使用方头斜字体。

(11) FontName属性。该属性的取值是控件对象标题等使用字体的字库名,必须是系统支持的各种字库。缺省字库是系统的缺省字库。

(12) FontSize属性。该属性的取值是数值,它定义控件对象标题等字体的字号。字号单位由FontUnits属性值定义。缺省值与系统有关。

(13) FontUnits属性。该属性的取值是points(磅,缺省值)、normalizedized(相对单位)、inches(英寸)、centimeters(厘米)或pixels(像素),该属性定义字号单位。相对单位将FontSize属性值解释为控件对象图标高度百分比,其他单位都是绝对单位。

(14) FontWeight属性。该属性的取值是normalized(缺省值)、light、demi或bold,它定义字体的粗细。

(15) HorizontalAlignment属性。该属性的取值是left、center(缺省值)或right。用来决定控件对象说明文字在水平方向上的对齐方式,即说明文字在控件对象图标上居左(left)、居中(center)、居右(right)。

3. 建立控件对象示例

(1) 建立按钮对象,当单击该按钮时绘制出正弦曲线。同时建立双位按钮,用于控制是否给坐标加网格线。

```
pbstart = uicontrol(gcf,'Style','push','Position',...
   [20,20,100,25],'String','Start Plot',...
   'CallBack','t = -pi:pi/20:pi;plot(t,sin(t))');
ptgrid = uicontrol(gcf,'Style','toggle','Position',...
   [150,20,100,25],'String','Grid','CallBack','grid');
```

Style属性值push指明该控件对象是按钮,toggle指明该控件对象是双位按钮, Position指示建立的按钮对象在当前的图形窗口中的位置及大小,String的属性值就是对象上的说明文字,CallBack属性定义了当用户单击该按钮对象时应执行的操作。

(2) 建立单选按钮,用来设置图形窗口的颜色,只能选择一种颜色。

```
htxt = uicontrol(gcf,'Style','text','String',...
   'Color Options','Position',[200,130,150,20]);
%建立单选按钮
hr = uicontrol(gcf,'Style','radio','String',...
   'Red','Position',[200,100,150,25],'Value',1,...
```

```
  'CallBack',['set(hr,"Value",1);','set(hb,"Value",0);',...
  'set(hy,"Value",0);','set(gcf,"Color","R")']);
hb = uicontrol(gcf,'Style','radio','String',...
  'Blue','Position',[200,75,150,25],...
  'CallBack',['set(hb,"Value",1),','set(hr,"Value",0);',...
  'set(hy,"Value",0);','set(gcf,"Color","B")']);
hy = uicontrol(gcf,'Style','radio','String',...
  'Yellow','Position',[200,50,150,25],...
  'CallBack',['set(hy,"Value",1),','set(hr,"Value",0);',...
  'set(hb,"Value",0);','set(gcf,"Color","Y")']);
```

Callback 执行的结果保证只有一个单选按钮的状态为 on,因为单选按钮的 Value 属性是这样定义的:如果单选按钮的状态是 on,那么属性 Value 的值是单选按钮的另一个属性 Max 的属性值,该属性值的缺省值是 1;如果状态是 off,那么属性 Value 的值是单选按钮的另一个属性 Min 的属性值,它的缺省值是 0。这样,通过对属性 Value 设定不同的值,就可以得到单选按钮的不同状态。

(3) 建立复选框,用于设置图形窗口的某些属性,如大小、颜色、标题等。

```
htxt = uicontrol(gcf,'Style','text','Position',[.1,.5,.25,.1],...
  'Units','normalized','String','Set Windows Properties');
hp = uicontrol(gcf,'Style','check','Position',...
  [.1,.4,.25,.1],'Units','normalized','String','MyPosition',...
  'CallBack',['set(gcf,"Position",[10,10,300,250]);',...
  'if get(hp,"Value") = = 1,',...
  'set(gcf,"Position",[10,10,600,500]),','end']);
hc = uicontrol(gcf,'Style','check','Position',...
  [.1,.3,.25,.1],'Units','normalized','String','MyColor',...
  'CallBack',['set(gcf,"color","w");',...
  'if get(hc,"Value") = = 1,','set(gcf,"color","g"),','end']);
hn = uicontrol(gcf,'Style','check','Position',...
  [.1,.2,.25,.1],'Units','normalized','String','MyName',...
  'CallBack',['set(gcf,"Name","复选框未选中");',...
  'if get(hn,"Value") = = 1,',...
  'set(gcf,"Name","复选框被选中"),','end']);
```

(4) 建立弹出框,其列表中包含一组可供选择的颜色。当选择某种颜色时,就将图形窗口的背景色设置为该颜色。

弹出框可选项在 String 属性中设置,每项之间用竖线字符"|"隔开,并用单撇号将所有的选项括起来。Value 属性的值是弹出式列表中选项的序号。例如,如果用户选列表中的第 2 项,那么 Value 的属性值就是 2。

```
hpop = uicontrol(gcf,'Style','popup','String',...
  'red|blue|green|yellow','Position',[100,100,100,80],...
  'CallBack',['cbcol = ["R","B","G","Y"];',...
  'set(gcf,"Color",cbcol(get(hpop,"Value")))']);
```

(5) 建立列表框,其作用与(4)同。

```
hl = uicontrol(gcf,'Style','list',...
    'String','red|blue|green|yellow|white|black',...
    'Position',[100,100,100,80],'Callback',...
    ['cbcol = {"r","b","g","y","w","k"};',...
    'set(gcf,"color",cbcol(get(hl,"value")));']);
```

(6) 建立一个编辑框,并加边框。

当需要用边框组织控件对象时,必须在定义控件对象之前建立边框对象,或者说边框对象必须覆盖该组中所有的控件对象。为了留出边框的边界,边框对象所占用的区域至少要比该组中所有控件对象占用的区域大。

编辑框分多行编辑框与单行编辑框。属性 Max 与属性 Min 之差小于或等于 1 时,为单行编辑框,否则为多行编辑框。

```
ftdir = uicontrol(gcf,'Style','frame',...
    'back','y','Position',[30,180,120,100]);
edmulti = uicontrol(gcf,'Style','edit',...
 'String','MATLAB is a very useful language.',...
 'Position',[50,200,75,55],'Max',2,'back','w');
```

在下列操作之后,MATLAB 将执行编辑框对象 Callback 属性定义的操作:

① 改变编辑框中的输入值,并将鼠标移出编辑框控件对象。

② 对单行编辑框,按下回车键,而不论编辑框中的值是否被改变。

③ 对单行和多行编辑框对象,按住 Ctrl 键,再按回车键,而不论编辑框中的值是否被改变。在多行编辑框中,按下回车键,可以输入下一行字符。

(7) 建立两个滑动条,分别用于设置图形窗口的宽度和高度,并利用静态文本说明对象,标出滑动条的数值范围以及当前值。

```
fig = figure('Position',[20,20,400,300]);
hsli1 = uicontrol(fig,'Style','slider','Position',...
  [50,50,120,20],'Min',200,'Max',800,'Value',400,...
  'CallBack',['set(azmcur,"String",'...
      'num2str(get(hsli1,"Value")));'...
      'set(gcf,"Position",[20,20,get(hsli1,"Value"),300]);']);
hsli2 = uicontrol(fig,'Style','slider','Position',...
  [240,50,120,20],'Min',100,'Max',600,'Value',300,...
  'CallBack',['set(elvcur,"String",'...
      'num2str(get(hsli2,"Value")));',...
      'set(gcf,"Position",[20,20,400,get(hsli2,"Value");])']);
%用静态文本标出最小值
azmmin = uicontrol(fig,'Style','text','Position',...
  [20,50,30,20],'String',num2str(get(hsli1,'Min')));
elvmin = uicontrol(fig,'Style','text','Position',...
  [210,50,30,20],'String',num2str(get(hsli2,'Min')));
%用静态文本标出最大值
azmmax = uicontrol(fig,'Style','text','Position',...
```

```
    [170,50,30,20],'String',num2str(get(hsli1,'Max')));
elvmax = uicontrol(fig,'Style','text','Position',...
    [360,50,30,20],'String',num2str(get(hsli2,'Max')));
%用静态文本标出当前设置的宽度和高度
azmLabel = uicontrol(fig,'Style','text','Position',...
    [50,80,65,20],'String','Width');
elvLabel = uicontrol(fig,'Style','text','Position',...
    [240,80,65,20],'String','Height');
azmcur = uicontrol(fig,'Style','text','Position',...
    [120,80,50,20],'String',num2str(get(hsli1,'Value')));
elvcur = uicontrol(fig,'Style','text','Position',...
    [310,80,50,20],'String',num2str(get(hsli2,'Value')));
```

例 7.3　建立如图 7.3 所示的数制转换对话框。在左边输入一个十进制整数和 2～16 之间的数，单击"转换"按钮能在右边得到十进制数所对应的 2～16 进制字符串，单击"退出"按钮退出对话框。



图 7.3　数制转换对话框

程序如下：

```
hf = figure('Color',[0,1,1],'Position',[100,200,400,200],...
    'Name','数制转换','NumberTitle','off','MenuBar','none');
uicontrol(hf,'Style','Text','Units','normalized',...
    'Position',[0.05,0.8,0.45,0.1],'Horizontal','center',...
    'String','输 入 框','Back',[0,1,1]);
uicontrol(hf,'Style','Text','Position',[0.5,0.8,0.45,0.1],...
    'Units','normalized','Horizontal','center',...
    'String','输 出 框','Back',[0,1,1]);
uicontrol(hf,'Style','Frame','Position',[0.04,0.33,0.45,0.45],...
    'Units','normalized','Back',[1,1,0]);
uicontrol(hf,'Style','Text','Position',[0.05,0.6,0.25,0.1],...
    'Units','normalized','Horizontal','center',...
```

```
    'String','十进制数','Back',[1,1,0]);
uicontrol(hf,'Style','Text','Position',[0.05,0.4,0.25,0.1],...
    'Units','normalized','Horizontal','center',...
    'String','2 - 16 进制','Back',[1,1,0]);
he1 = uicontrol(hf,'Style','Edit','Position',[0.25,0.6,0.2,0.1],...
    'Units','normalized','Back',[0,1,0]);
he2 = uicontrol(hf,'Style','Edit','Position',[0.25,0.4,0.2,0.1],...
    'Units','normalized','Back',[0,1,0]);
uicontrol(hf,'Style','Frame','Position',[0.52,0.33,0.45,0.45],...
    'Units','normalized','Back',[1,1,0]);
ht = uicontrol(hf,'Style','Text','Position',[0.6,0.5,0.3,0.1],...
    'Units','normalized','Horizontal','center','Back',[0,1,0]);
COMM = ['n = str2num(get(he1,''String''));','b = str2num(get(he2,''String''));',...
    'dec = trdec(n,b);','set(ht,''string'',dec);'];
uicontrol(hf,'Style','Push','Position',[0.18,0.1,0.2,0.12],...
    'String','转 换','Units','normalized','Call',COMM);
uicontrol(hf,'Style','Push','Position',[0.65,0.1,0.2,0.12],...
    'String','退 出','Units','normalized','Call','close(hf)');
```

程序调用了 trdec.m 函数文件，该函数的作用是将任意十进制整数转换为 2～16 进制字符串。trdec.m 函数文件如下：

```
function dec = trdec(n,b)
ch1 = '0123456789ABCDEF';             %十六进制的 16 个符号
k = 1;
while n ~ = 0                         %不断除某进制基数取余直到商为 0
    p(k) = rem(n,b);
    n = fix(n/b);
    k = k + 1;
end
k = k - 1;
strdec = '';
while k > = 1                         %形成某进制数的字符串
    kb = p(k);
    strdec = strcat(strdec,ch1(kb + 1:kb + 1));
    k = k - 1;
end
dec = strdec;
```

例 7.4　建立如图 7.4 所示的图形演示对话框。在编辑框输入绘图命令，单击“绘图”按钮能在左边坐标轴得到所对应的图形，弹出框提供色图控制，列表框提供坐标网格线和坐标边框控制。

程序如下：

```
clf;
set(gcf,'Unit','normalized','Position',[0.2,0.3,0.65,0.35]);
```

```
set(gcf,'Menubar','none','Name','图形演示','NumberTitle','off');
axes('Position',[0.05,0.15,0.55,0.7]);
uicontrol(gcf,'Style','text', 'Unit','normalized',...
   'Posi',[0.63,0.85,0.2,0.1],'String','输入绘图命令','Horizontal','center');
hedit = uicontrol(gcf,'Style','edit','Unit','normalized','Posi',[0.63,0.15,0.2,0.68],...
   'Max',2);            % Max 取 2,使 Max - Min > 1,从而允许多行输入
hpopup = uicontrol(gcf,'Style','popup','Unit','normalized',...
   'Posi',[0.85,0.8,0.15,0.15],'String','Spring|Summer|Autumn|Winter');
hlist = uicontrol(gcf,'Style','list','Unit','normalized',...
   'Posi',[0.85,0.55,0.15,0.25],'String','Grid on|Grid off|Box on|Box off');
hpush1 = uicontrol(gcf,'Style','push','Unit','normalized',...
   'Posi',[0.85,0.35,0.15,0.15],'String','绘 图');
uicontrol(gcf,'Style','push','Unit','normalized',...
   'Posi',[0.85,0.15,0.15,0.15],'String','关 闭','Call','close all');
set(hpush1,'Call','COMM(hedit,hpopup,hlist)');
set(hlist,'Call','COMM(hedit,hpopup,hlist)');
set(hpopup,'Call','COMM(hedit,hpopup,hlist)');
```

COMM.m 函数文件:

```
function COMM(hedit,hpopup,hlist)
com = get(hedit,'String');
n1 = get(hpopup,'Value');
n2 = get(hlist,'Value');
if ~ isempty(com)          %编辑框输入非空时
    eval(com');            %执行从编辑框输入的命令
    chpop = {'spring','summer','autumn','winter'};
    chlist = {'grid on','grid off','box on','box off'};
    colormap(eval(chpop{n1}));
    eval(chlist{n2});
end
```

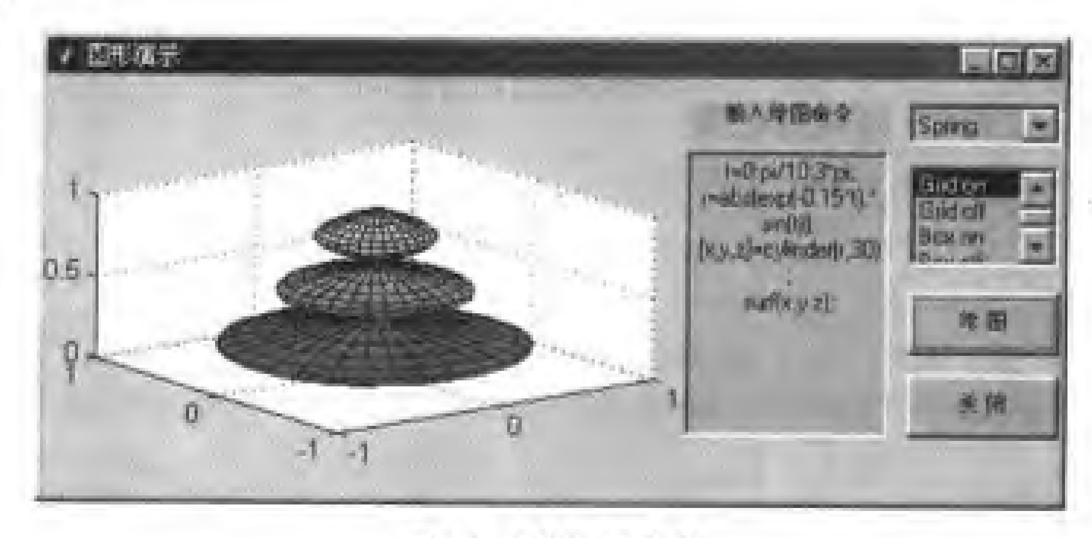


图 7.4 图形演示对话框

7.3 用户界面设计工具

前面介绍了用于用户界面设计的有关函数，为了更方便快捷地进行用户界面设计，MATLAB提供了用户界面设计工具，利用这些工具使得界面设计过程变得简单和直接，实现"所见即所得"。MATLAB的用户界面设计工具共有5个，它们是：图形界面控制面板、属性编辑器、事件过程编辑器、菜单编辑器和位置调整工具。下面简要介绍各种工具的使用方法。

7.3.1 图形界面控制面板

图形界面控制面板是MATLAB用户界面设计工具的集成环境。在MATLAB命令窗口中输入命令guide或在MATLAB命令窗口File菜单中选择Show GUI Layout Tool命令，将启动如图7.5所示的图形界面控制面板。图形界面控制面板分为3个部分：界面编辑工具栏、受控图形窗口列表框和控件对象创建工具栏。

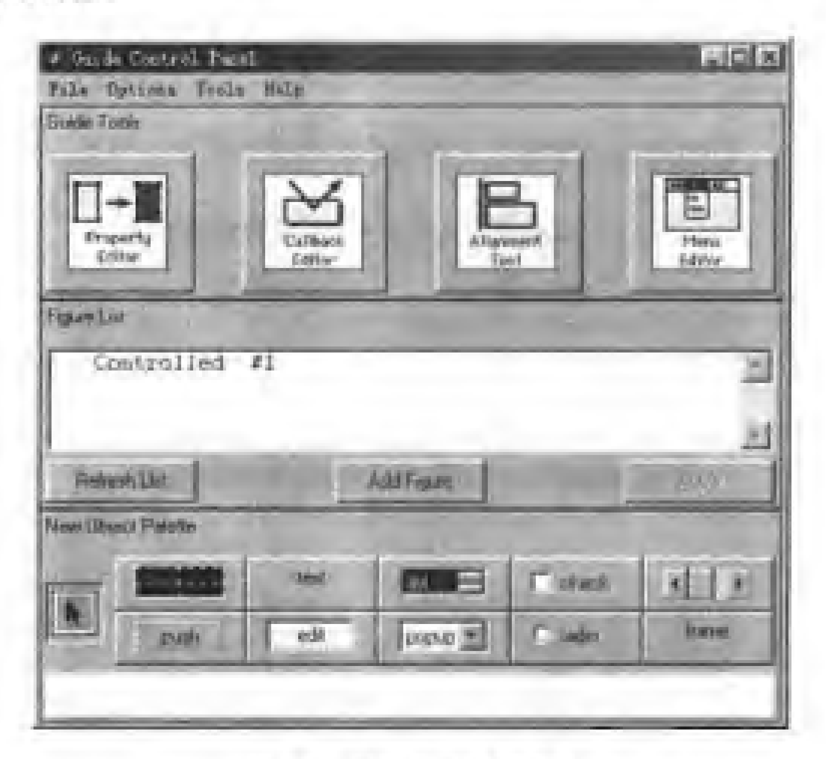


图7.5 图形界面控制面板

第一排是界面编辑工具栏，其中共有属性编辑器、事件过程编辑器、位置调整工具和菜单编辑器4个工具。要打开某一编辑工具，只需单击相应的按钮即可。

第二排是受控图形窗口列表框，其中列出了当前打开的图形窗口，并显示窗口的受控状态。相对于图形界面控制面板，所有的图形窗口只有两种状态：受控状态和激活状态。当窗口处于受控状态时，可以编辑窗口中的所有图形对象。从这个意义上讲，受控状态又可称作编辑状态。激活状态是通常意义下的状态，当窗口处于激活状态时，用户能够对其进行正常的操作，例如按钮可以按下、弹出框可以打开等。刚启动图形界面控制面板时，当前图形窗口将自动处于受控状

态。为了使某个图形窗口处于受控状态，只要在图形窗口列表框中单击相应的图形窗口名字，再单击 Apply 按钮即可。

第三排是控件对象创建工具栏，利用这些工具可以很方便地在图形窗口中添加各种控件对象和坐标系对象。

7.3.2 属性编辑器

属性编辑器用于对图形对象属性进行操作和管理，其功能与 set 函数和 get 函数相同，但属性编辑器更容易操作和使用。

启动属性编辑器有 3 种方法：在 MATLAB 命令窗口输入 propedit 命令或在图形界面控制面板中启动属性编辑器，还可以在命令窗口 File 菜单中选择 Show Graphics Property Editor。启动属性编辑器后，将打开属性编辑器窗口，在该窗口可进行有关编辑操作。例如，下列命令先绘制出正弦曲线并建立一个图形窗口，然后打开属性编辑器：

```
t=0:0.1:20;
plot(t,sin(t));
propedit(gcf);
```

打开的属性编辑器如图 7.6 所示。菜单栏下面为图形对象列表框，该列表框下有 2 个编辑



图 7.6 属性编辑器

框，左边的一个显示属性名，右边的一个显示属性值。只要在其中分别输入属性名和属性值，就可以修改图形对象的属性。例如，为了将图形窗口的背景颜色改为红色，可以在属性名编辑框输入属性名 Color(不要单撇号)，按回车后，Color 属性的缺省值就出现在属性值编辑框，将该缺省值改为 red(带单撇号)或[1 0 0]，回车后，相应的图形窗口背景颜色就变为红色。这等价于在命令

窗口使用 set(gcf,'Color',[1 0 0])命令。

由于图形对象的属性很多，所以在修改图形对象属性时，直接输入属性名是很不方便的。为了避免这一点，属性编辑器提供了属性列表框。在属性编辑器窗口中，选中 Show Property List 复选框，将打开属性列表框。在属性列表框中列出了被编辑图形对象的全部属性和当前的属性值。为了编辑某个属性，只要单击相应的属性项，则相应的属性名和属性值就会出现在编辑框中，然后改变属性值即可。

可以用 Show Object Browser 复选框打开图形对象列表框。在图形对象列表框中会列出当前图形窗口中的图形对象及其层次关系。只要在图形对象列表框中选择某一图形对象，即可将属性编辑器窗口对准到该对象，再对其属性进行必要的编辑修改。如果此时属性列表框是打开的，那么该图形对象的所有属性就在属性列表框中列出。

此外，属性编辑器窗口还具有以下功能：

(1) 在属性编辑器中，可以同时选择多个图形对象，甚至可以是不同类型的图形对象。此时，在属性列表框中只显示所有被选中对象的共有属性，通过设置某个属性的值，所有被选中的图形对象都将更改为同样的值。

(2) 在属性编辑器中，允许输入属性名的前几个字母来代表整个属性名。如在属性名编辑框输入 Col 即可以代替 Color 属性名。

(3) 在属性名编辑框中，属性编辑器忽略含有空格的输入。所以如果输入了错误的属性名，只要按空格再输入正确的属性名即可。

7.3.3 事件过程编辑器

用事件过程编辑器可以编辑修改某个图形对象的 Callback 属性值。可以在命令窗口用 cbedit 命令启动事件过程编辑器，也可以从图形界面控制面板启动事件过程编辑器。事件过程编辑器如图 7.7 所示。



图 7.7 事件过程编辑器

用事件过程编辑器编辑 Callback 属性值时，首先要选中 Show Object Browser 复选框，然后选中想要修改属性的对象，再在属性弹出框中选中相应的属性，接着在编辑框中输入相应的命令代码，最后单击 Apply 按钮。

7.3.4 菜单编辑器

菜单编辑器既可以在某个图形窗口的菜单条上添加菜单项，又可以编辑已定义的菜单项。有一点要注意，只有图形窗口处于激活状态时，用菜单编辑器添加或修改的菜单项才会出现在图形窗口的菜单条上。

在命令窗口输入 menuedit 命令或从图形界面控制面板单击菜单编辑器按钮都能启动菜单编辑器。菜单编辑器如图 7.8 所示，其使用方法与其他工具相似。

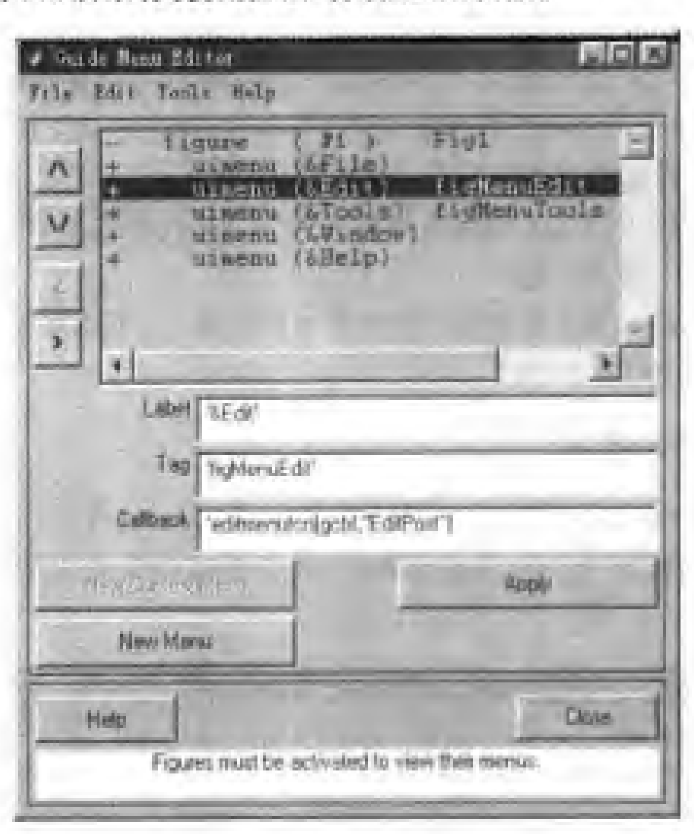


图 7.8 菜单编辑器

7.3.5 位置调整工具

位置调整工具主要用于对图形对象的几何位置进行调整，使得界面更加美观。使用命令进行用户界面设计时，需要仔细计算每个图形对象在图形窗口中的位置坐标，且不能直接面对所设计的界面，因此是非常麻烦的。利用位置调整工具就相当简单。

在命令窗口输入 align 命令或从图形界面控制面板单击位置调整工具按钮都能启动位置调整工具。位置调整工具如图 7.9 所示，它由对象列表框和调整工具按钮组成。在选择了图形对象后，就可以用各种工具按钮来移动被选中的图形对象，或调整它与其他图形对象的相对位置。

以上介绍了 MATLAB 的用户界面设计工具，下面给出一个例子，以说明这些工具的具体运用。

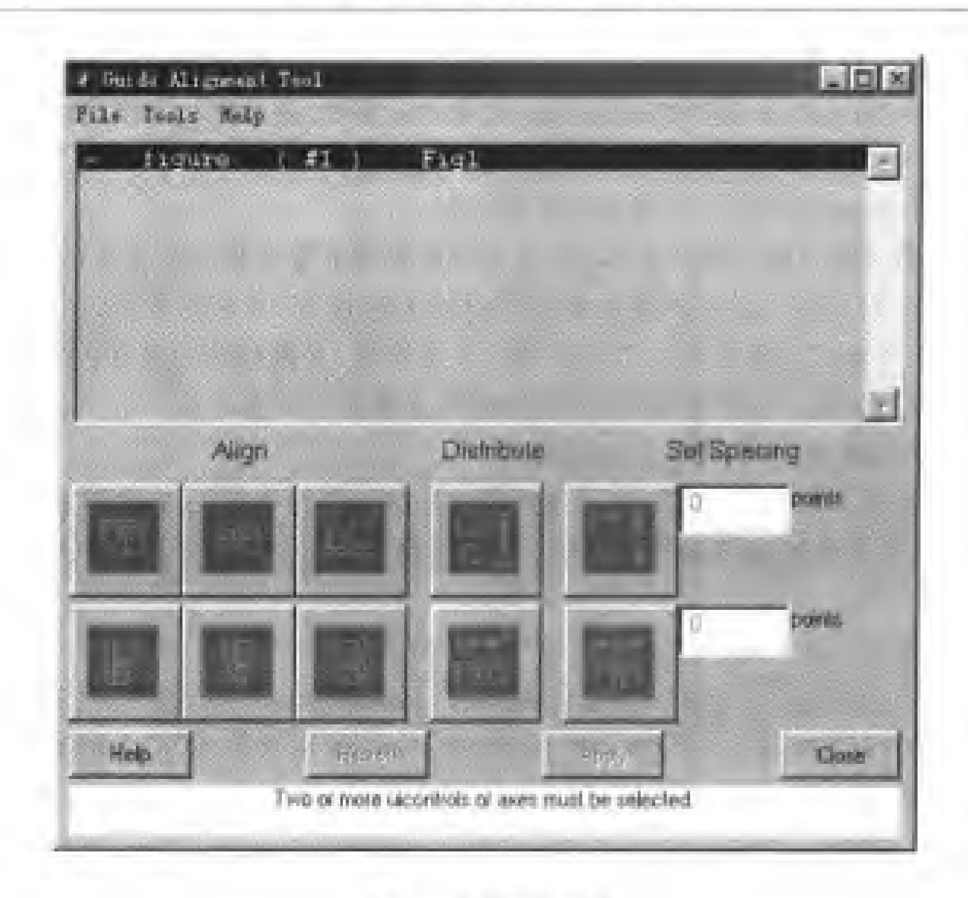


图7.9 位置调整工具

例7.5 利用界面设计工具设计图7.10所示的用户界面。具体要求是：

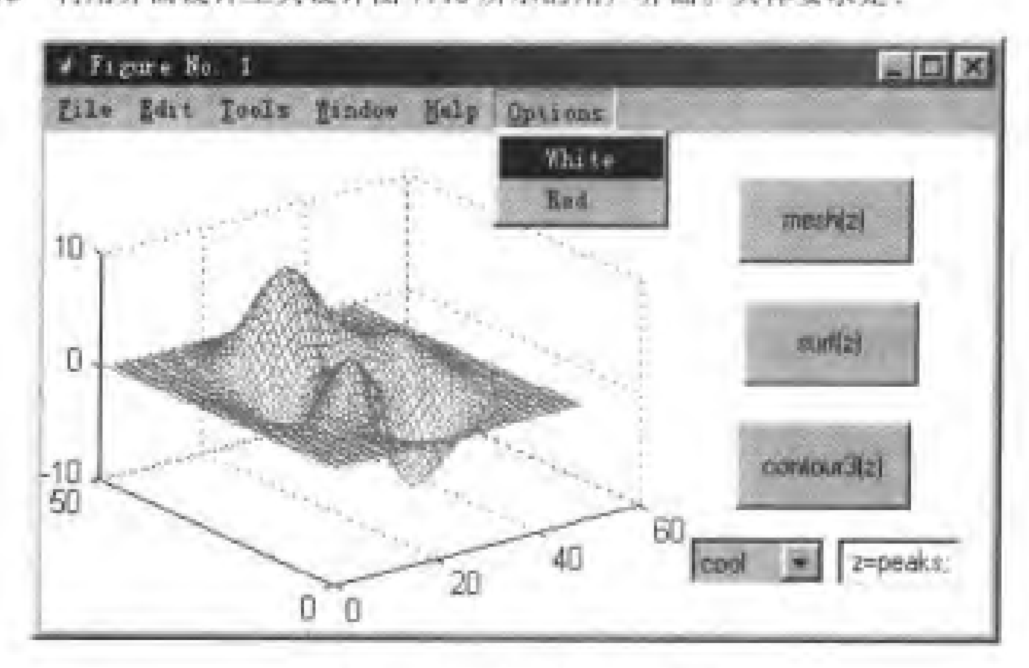


图7.10 利用工具设计用户界面

(1) 在编辑框输入形成数据 z 的命令，然后选择3个按钮中的一个，即可绘制出 z 的图形。

(2) 选择弹出框中的某个色图，能及时地更改绘图的颜色。

(3) 在图形窗口缺省的菜单条上添加一个菜单项 Options，Options 下又有两个子菜单项 White 和 Red，选中 White 项时，图形窗口将变成白色，选中 Red 项时，图形窗口将变成红色。

操作步骤如下：

(1) 打开图形界面控制面板，添加有关图形对象

在 MATLAB 命令窗口输入命令 guide，将打开图形界面控制面板和一个新的图形窗口，该图形窗口处于受控状态。单击图形界面控制面板中的 axes 按钮，并在图形窗口中拖出一个矩形框，调整好大小和位置。再添加一个弹出框和一个编辑框，并调整好大小和位置。添加 3 个按钮，并调整好大小和位置。必要时可利用位置调整工具将图形对象对齐。

(2) 利用属性编辑器，设置图形对象的属性

打开属性编辑器，选中 3 个按钮，利用属性编辑器把它们的 Position 属性的第三、第四个分量设为相同的值，以使 3 个按钮宽和高都相等。3 个按钮的 String 属性分别是说明文字 mesh(z)、surf(z)和 contour3(z)。

为设置弹出框的 String 属性，先在命令窗口输入命令：

```
cmap = {'hot';'hsv';'gray';'cool';'prism';'winter';'summer'};
```

这里列出了 MATLAB 提供的几个色图。把弹出框的 String 属性设为 cmap(不用加撇号)。

利用事件过程编辑器，把弹出框的 Callback 属性设为：

```
val = get(gcbo,'Value');
str = get(gcbo,'String');
colormap(str{val})
```

其中第一条命令获得弹出框的 Value 属性值，第二条命令获得弹出框的 String 属性值，第三条命令选择相应的色图。gcbo 获取当前回调对象的句柄(get current callback object)。

把 3 个按钮的 Callback 属性设置为：

```
comstr = get(gcbo,'String');
edith = findobj(gcbf,'Tag','EditText1');
zstr = get(edith,'String');
eval(zstr);
eval(comstr);
```

其中第一条命令获得按钮的 String 属性值，即绘图命令；第二条命令获得 Tag 属性值为 EditText1 的对象的句柄值，即编辑框的句柄值；第三条命令获得编辑框中的字符串；第四条命令运行编辑框中的字符串代表的命令；第五条命令运行绘图命令。gcbf 获取当前回调图形窗口的句柄(get current callback figure)。

(3) 添加 Options 菜单项

打开菜单编辑器，选中图形窗口对象，并单击 New Menu 按钮，把新建菜单项的 Label 属性设为 Options，再单击 Apply 按钮。这样将在图形窗口菜单条上增加一个新的菜单项 Options。

接下来在刚建的 Options 菜单项下建立子菜单项。选中 Options，单击 New Menu 按钮，把子菜单项的 Label 属性设为 White，把 Callback 属性设为 set(gcbf,"Color","w")。同理，再为 Options 建立一个子菜单项，Label 属性和 Callback 属性分别设为 Red 和 set(gcbf,"Color","r")。

至此，图形用户界面已经设计好。在图形窗口选择 File 菜单项中的 Save Figure As 或 Save

Figure 命令用 M 文件把图形存起来，然后在 MATLAB 命令窗口运行该 M 文件即可得到图 7.10 所示的图形用户界面。

习　题　七

1. 什么是图形用户界面？有何特点？

2. 菜单设计和对话框设计的基本思路是什么？

3. 对话框中常用控件有哪些？各有何作用？

4. 设计菜单。菜单条中含有 File 和 Help 两个菜单项。如果选择 File 中的 New 选项，则将显示 New Item 字样；如果选择 File 中的 Open 选项，则将显示出 Open Item 字样。File 中的 Save 菜单项初始时处于禁选状态，在选择 Help 选项之后将此菜单项恢复成可选状态，如果选择 File 中的 Save 选项，则将显示 Save Item 字样；如果选择 File 中的 Exit 选项，则将关闭当前窗口。如果选择 Help 中 About …选项，则将显示 Help Item 字样，并将 Save 菜单设置成可选状态。

5. 建立控件对象。

(1) 建立单选按钮，分别用于将图形窗口移至屏幕的 4 个角。

(2) 建立弹出框，分别选择不同的函数，从而实现相应的函数运算。

(3) 建立列表框，分别选择不同的函数，从而实现相应的函数运算.

(4) 分别建立编辑框和命令按钮，其中编辑框输入多项式系数，命令按钮求其根。

(5) 用滑动条来输入 a 和 b 的值，命令按钮求其和。

(6) 在图形窗口中央建立一个按钮，单击按钮时，按钮在图形窗口中随机游动。

(7) 建立一个双位按钮，控制是否保留坐标轴原有图形。

6. 设计一个“MATLAB 功能演示系统”，要求系统具有友好的用户界面。

7. 利用用户界面设计工具设计图 7.4 所示的图形演示对话框。

第 8 章　MATLAB 笔记本

MATLAB 和文字处理软件 Microsoft Word 的结合，使用户能够在 Word 环境中访问 MATLAB，为用户提供一个集文字处理与科学计算于一体的工作环境。这种工作环境称为 MATLAB 笔记本（Notebook）。MATLAB 笔记本不仅拥有 Word 的全部功能，而且能输入并执行 MATLAB 命令，从而能将命令执行结果直接插入在 Word 文档中。

在 MATLAB 笔记本中，文档、图形、表格及数学公式等的输入、编辑、排版方法与常规的 Word 操作完全相同。关于 Word 的操作，不再赘述。本章重点介绍如何利用 MATLAB 笔记本在 Word 文档中输入、执行 MATLAB 命令以及保存命令执行结果。

8.1　笔记本的安装及启动

8.1.1　笔记本的安装

随 MATLAB 版本的不同，笔记本的安装方法也不同。对于 MATLAB 5.0～5.2 版，笔记本的安装是在 MATLAB 安装过程中进行的，而 MATLAB 5.3 版笔记本的安装是在 MATLAB 安装完成之后，在 MATLAB 环境下进行的。下面介绍 MATLAB 5.3 版笔记本的安装。具体步骤为：

(1) 启动 MATLAB，在其命令窗口输入：

```
notebook - setup
```

于是屏幕提示：

```
Welcome to the utility for setting up the MATLAB Notebook
for interfacing MATLAB to Microsoft Word
Choose your version of Microsoft Word:
[1] Microsoft Word for Windows 95 (Version 7.0)
[2] Microsoft Word 97
[3] Exit, making no changes
Microsoft Word Version:
```

(2) 用户根据自己机器上所用 Word 版本，在最后一行提示后而输入对应序号，并按回车键。于是出现以下提示：

```
You will be presented with a dialog box. Please use it to select
your copy of the Microsoft Word 95 executable (winword.exe).
Press any key to continue...
```

(3) 当按下任意键后，屏幕上会弹出一个对话框，让用户选择 winword.exe 所在的目录。当选定并确认后，又出现下面的提示：

```
You will be presented with a dialog box. Please use it to
```

```
select a Microsoft Word template (.dot) file in one of your
Microsoft Word template directories. We suggest that you specify
your normal.dot file.
Press any key to continue...
```

(4) 当按下任意键后,屏幕上又弹出一个对话框,让用户选择 normal.dot 所在的目录。当选定并确认后,出现提示:

```
Notebook setup is complete.
```

表示笔记本安装结束。

8.1.2　笔记本的启动

启动笔记本有两种方法:从 Word 中启动;从 MATLAB 命令窗口启动。

1. 从 Word 中启动笔记本

启动 Word 后,一般按系统默认的模板 normal.dot 建立文档,文档中可以包括任意文本,当然也可以包括 MATLAB 命令。但若想使 MATLAB 命令在 Word 下执行并将执行结果也包含在文档中,则必须打开 MATLAB 命令窗口,在其中运行该命令,然后把运行结果剪贴到该文档中。显然,这是非常麻烦的。而采用 M-book 模板,则可以使上述操作直接在 Word 下进行。所谓从 Word 中启动笔记本,实际上就是新建或打开一个业已存在的 M-book 文档。

(1) 建立新的 M-book 文档

先启动 Word,假若这时 Word 的默认模板是 normal.dot。在这种情况下,建立新的 M-book 文档的步骤是:从 Word 窗口的"文件"菜单项中选择"新建"命令,在弹出的对话框中选择 M-book 模板,单击"确定"按钮。于是,Word 窗口由原先的默认式样变成 M-book 式样。假如此前 MATLAB 尚未启动,则 MATLAB 会自行启动,用户可以看到 MATLAB 的启动图标。MATLAB 启动结束后,便进入新的 M-book 文档。

(2) 打开已有的 M-book 文档

打开已有的 M-book 文档与打开一般 Word 文档没有什么区别。最常用的方法是:从 Word 窗口的"文件"菜单项中选择"打开"命令,然后从弹出的对话框中,选择需要的 M-book 文档。

2. 从 MATLAB 中启动笔记本

从 MATLAB 中启动笔记本比较简单,只需在命令窗口中键入命令:

notebook 或 notebook 文件名

不带文件名的 notebook 命令用于建立一个新的 M-book 文档,带文件名的 notebook 命令用于打开一个已存在的 M-book 文档。

8.1.3　MATLAB 笔记本的界面

M-book 模板为用户提供了在 Word 环境下使用 MATLAB 的功能。该模板定义了 Word 与 MATLAB 进行通信的宏指令、文档样式和工具栏。当调用该模板时,Word 便自动将 M-book 模板定义的宏装入内存,启动笔记本用户界面,以及定义文档的样式和单元。

MATLAB 笔记本的界面和通常的 Word 界面主要有两点区别:

(1) 在菜单栏中多了一个 Notebook 菜单项,笔记本的许多操作都可以通过该菜单项的命令

来完成。

(2) 在"文件"菜单项下多了一个 New M - book 命令项。如果在 M - book 模板下要建立新的 M - book 文档,可以选择该命令。

8.2 输入单元的定义与执行

在 MATLAB 笔记本中,凡在 Word 与 MATLAB 之间进行传递的内容称为单元(Cell)。由 M - book 文档传向 MATLAB 的命令称为输入单元(Input Cell)。文档中任何合法的 MATLAB 命令都可以定义为输入单元,输入单元可以送到 MATLAB 环境中去执行,执行结果保存在 MATLAB 工作空间,同时送回笔记本,成为输出单元(Output Cell)。输入单元可以单独存在,但输出单元必须依赖输入单元而存在。较之通常的 Word,MATLAB 笔记本最重要的操作就是输入单元的定义与执行。

8.2.1 基本操作

1. 输入单元的定义

在 M - book 中,输入单元可以是一条单独的 MATLAB 命令、在一行内的几条 MATLAB 命令、多行命令或者是内嵌在文本中的命令。定义输入单元的方法是:首先选中所需命令,然后在 Notebook 菜单项中选择 Define Input Cell 命令,于是被选中的 MATLAB 命令成为输入单元。输入单元在 M - book 模板中预设置为绿色,10 磅大小,Courier New 粗体英文。定义输入单元也可以在选中所需命令后,直接按组合键 Alt + D。

要注意,定义的输入单元并没有执行,当然也就没有执行结果。

2. 输入单元的执行

为了执行输入单元,应选择 Notebook 菜单项中的 Evaluate Cell 命令或直接按组合键 Ctrl + Enter。

3. 输出单元

输入单元执行后产生输出单元。如果输入单元经修改后重新执行,那么新的输出单元将替换原有的输出单元。输出单元中包含 MATLAB 命令的输出结果:数据、图形、错误信息。数据的输出样式在 M - book 模板中设置为蓝色,10 磅大小,Courier New 英文细体。错误信息的输出样式是红色,10 磅大小,Courier New 英文粗体。图形的输出格式则通过 Notebook 菜单中的 Notebook Options 来设置。

例 8.1 定义并执行输入单元。

操作步骤如下:

(1) 在文档中输入 MATLAB 命令:

```
x = (1:5)/5 * pi;y = 2 * exp( - 0.5 * x) . * sin(2 * pi * x)
```

(2) 选中命令行,在 Notebook 菜单项中选 Define Input Cell 命令或直接按组合键 Alt + D,于是命令行就变成了"绿色"的输入单元。

(3) 若要把输入单元送去执行,则可用 Notebook 菜单项中的 Evaluate Cell 命令或直接按组合

键 Ctrl + Enter,执行后产生“蓝色”的输出单元:

```
y =
-1.0543   1.0660   -0.5155   -0.0474   0.3230
```

在 M - book 文档中,输入 MATLAB 命令时要特别注意:

(1) 不要把中文标点混杂在 MATLAB 命令中,否则会产生错误。MATLAB 命令及其中的标点都必须在英文状态下输入。

(2) MATLAB 的续行符(3 个小黑点...),不能用于 M - book。M - book 处理超过物理行命令的方法是让其自行换行,不要加回车符。

4. 输入单元定义与执行同时进行

先选中 MATLAB 命令,然后从 Notebook 菜单项中选择 Evaluate Cell 命令或直接按组合键 Ctrl + Enter,不但使被选中的命令成为输入单元,而且送去执行,产生输出单元。

例 8.2 输入单元定义与执行同时进行。

在英文状态下,以文本方式键入命令,然后选中命令并按 Ctrl + Enter 键,则得到输入、输出单元:

```
z = y.^2, a = [x;y;z;x.*y;x./z], inv(a)
z =
    1.1115   1.1365   0.2658   0.0022   0.1043
a =
  1.0e+003 *
     0.0006   0.0013    0.0019    0.0025   0.0031
    -0.0011   0.0011   -0.0005   -0.0000   0.0003
     0.0011   0.0011    0.0003    0.0000   0.0001
    -0.0007   0.0013   -0.0010   -0.0001   0.0010
     0.0006   0.0011    0.0071    1.1177   0.0301
ans =
    -0.0656   -0.8691    0.3727    0.4369    0.0002
    -0.0097    0.7026    0.5241   -0.2481    0.0000
     0.2385    0.8841    0.1008   -1.0123   -0.0006
    -0.0068    0.0116    0.0081   -0.0105    0.0009
     0.1976   -0.6469   -0.3511    0.6276   -0.0004
```

说明:

(1) 例中变量 x,y 的值取自 MATLAB 工作空间,由例 8.1 的输出确定。

(2) 在笔记本中,是否显示计算结果的控制方法与在 MATLAB 命令窗口一样。命令后有分号,则计算结果不会以输出单元显示,命令后不带分号,则显示。

(3) 修改输入单元后再重新执行,则新的输出结果会覆盖掉原来的输出结果。

8.2.2 自初始化单元及其应用

1. 自初始化单元

自初始化单元(AutoInit Cell)与输入单元功能惟一不同的是:当用户启动一个 M - book 文档

时，包含在该文档中的自初始化单元会自动被送去执行，而输入单元不具备这种功能。若用户需要在打开文档时，对 MATLAB 工作空间进行初始化工作，那么自初始化单元特别有用。自初始化单元在 M - book 模板中预定义为深蓝色，10 磅大小，Courier New 英文粗体。

自初始化单元有两种来源：文本形式的 MATLAB 命令；已经存在的输入单元。为把它们变成自初始化单元，需先选中它们，然后选择 Notebook 菜单中的 Define AutoInit Cell 命令即可。

2. 自初始化单元的应用

M - book 的所有运算都是在 MATLAB 中进行的，参与运算的所有变量都储存在 MATLAB 工作空间中。各 M - book 文档和 MATLAB 命令窗口分享同一个“计算引擎”和同一个工作空间。工作空间中的变量是各 M - book 文档和 MATLAB 命令窗口工作后共同产生的。

当用户同时打开几个 M - book 文档，或者在 MATLAB 命令窗口和 M - book 文档之间交互运作时，应注意不同文档之间，或者文档与窗口之间变量的相互影响。为了使某一个 M - book 文档独占 MATLAB 工作空间，一个有效的办法是：把 clear 命令作为该文档的第一个自初始化单元。

8.2.3 单元群及其应用

1. 单元群

单元群(Cell Group)是多行输入单元或自初始化单元组成的一个整体。它有 3 种来源：

(1) 对输入的多行文本型 MATLAB 命令，用鼠标把它们同时选中，然后在 Notebook 菜单中选择 Define Cell 或 Define AutoInit Cell 命令，便生成输入单元群或自初始化单元群。

(2) 对输入的多行文本型 MATLAB 命令，用鼠标把它们同时选中，然后在 Notebook 菜单中选择 Evaluate Cell 或按组合键 Ctrl + Enter，于是单元群被定义并执行。

(3) 把已有的多个独立输入单元或自初始化单元同时选中，然后在 Notebook 菜单中选择 Group Cells，于是，便获得以第一个独立单元的性质组合而成的单元群。

假如被选中的多个独立单元区域内，夹带有独立成行的文本，那么在把独立单元组合成群的同时，将把原先夹在中间的文本内容移到单元群之后。

顺便指出：Notebook 菜单中的 Group Cells 命令不能直接把文本型 MATLAB 命令组合成单元群。单元群被激活后将拥有一个输出单元(群)。在这个输出单元(群)中，输出数据的次序与命令在(输入)单元群中的次序相同，而图形输出总在数据之后。

2. 单元群的应用

单元群的用途主要有两个：

(1) 保证 MATLAB 命令结构(如循环结构、条件结构)的完整。

(2) 保证输出结果(如图形)的完整。

例 8.3 对循环结构使用单元群。

```
clear
x = 0:10;
for k = 1:10
    y = k * x;
    plot(x,y);
    hold on
```

```
end
hold off
set(gcf,'Color','w')
```

执行结果如图 8.1 所示。假如本例中的单元群用一行行独立的输入单元替代，那么当运行它们时，将显示出错警告。

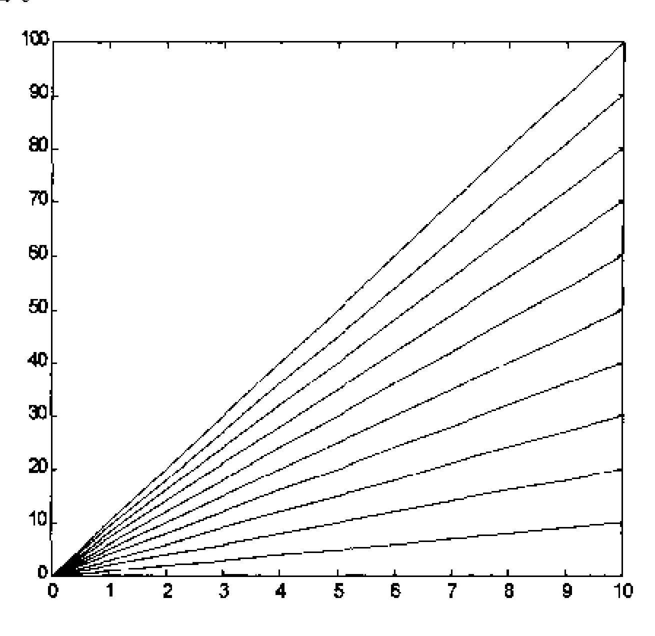


图 8.1 对循环结构使用单元群

例 8.4 使用单元群产生完整图形。

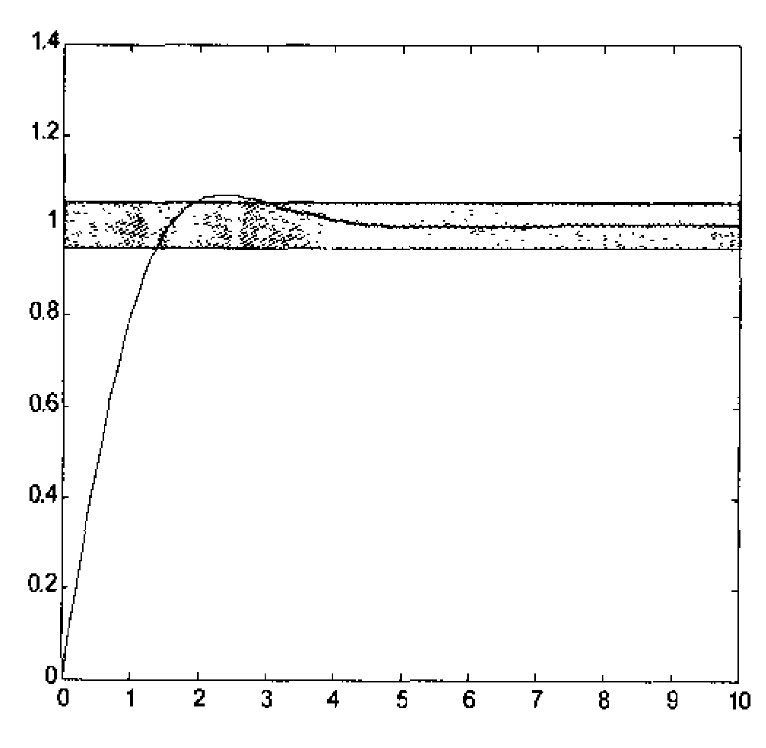


图 8.2 使用单元群产生的图形

```
clf;
t=0:0.1:10;y=1-exp(-t).*cos(t);
tt=[0,10,10,0];
yy=[0.95,0.95,1.05,1.05];
fill(tt,yy,'y');
hold on
plot(t,y,'k');
hold off
set(gcf,'Color','w')
ymax = max(y),tm = t(find(y = = ymax))
ymax =
    1.0669
tm =
    2.4000
```

执行结果如图 8.2 所示。

8.2.4 单元的循环执行

利用 Notebook 菜单中的 Evaluate Loop 命令可实现单元的循环执行。操作步骤如下：

(1) 选择要重复执行的输入单元，再选择 Notebook 菜单中的 Evaluate Loop 命令，弹出图 8.3 所示的循环执行对话框。

(2) 在对话框的 Stop After 栏中输入所需重复执行次数，然后单击 Start 按钮，开始执行命令，并显示已执行的次数。

(3) 如果要在每一次循环后加入延迟，可以单击 Slower 按钮，反之单击 Faster 按钮。如果要暂停命令的执行，可单击 Pause 按钮。

(4) 单击 Close 按钮，关闭对话框。

图 8.3　循环执行对话框

例 8.5　用单元的循环执行产生图 8.1。

操作步骤如下：

(1) 运行以下命令：



```
clear;x=0:10;k=1;hold on;
```

(2) 选中以下 3 行命令，再选择 Notebook 菜单中的 Evaluate Loop 命令打开循环执行对话框，取重复次数为 10 次，单击 Start 按钮，命令循环执行。

```
y=k*x;
plot(x,y);
k=k+1;
```

(3) 待循环结束，单击 Close 按钮，关闭对话框。

8.3　计算区的定义与执行

计算区(Calc Zone)是一个由普通 Word 文本、输入单元和输出单元组成的连续区，用于描述某个具体的作业或问题。在计算区里，用户可以根据描述问题的需要，安排段落、标题、分栏，而不受计算区外的有关格式的约束。

MATLAB 笔记本将计算区定义为 Word 的一个节，并将节的分隔符号置于计算区的首尾，但在文档的开始和结束的地方 Word 不显示节分隔符号。

定义计算区的方法是：先选定包含普通 Word 文本、输入单元和输出单元的一个连续区，然后选择 Notebook 菜单中的 Define Calc Zone 命令。

一旦计算区被定义后，不管光标在计算区的什么位置，只要选择 Notebook 菜单中的 Evaluate Calc Zone 命令即可执行计算区中的全部输入单元，且在每个输入单元后面以输出单元形式给出相应的计算结果。

8.4　输出格式控制

输出格式控制包括输出数据控制和输出图形控制。可以通过 Notebook 菜单中的 Notebook Options 命令来实现。选择该命令后，会弹出如图 8.4 所示的输出格式控制对话框，以后的操作都是针对该对话框进行的。

8.4.1　输出数据格式控制

1. 输出数据的表示方式

在 M - book 文档中，输出单元中数据的输出格式有 3 种控制方法：

(1) 在图 8.4 所示的对话框中，通过 Numeric Format 弹出式列表框进行设置，共有 8 种可选格式：Short、long、Hex、Bank、Plus、Short e、Long e、Rational。

(2) 通过输入单元中的 format 命令进行设置。

(3) 在 MATLAB 命令窗口中，用 format 命令进行设置。

在此再次说明，不同输出格式给出不同的数据显示精度，但内部存储及运算都是以相同的双精度进行的。

2. 输出数据间的空行控制

在图 8.4 所示的对话框中，选择 Loose 或 Compact，可以控制输入单元与输出单元之间有无空行。如果选择 Loose，则在输入单元与输出单元之间插入一个空行，而选择 Compact，不插入空行。

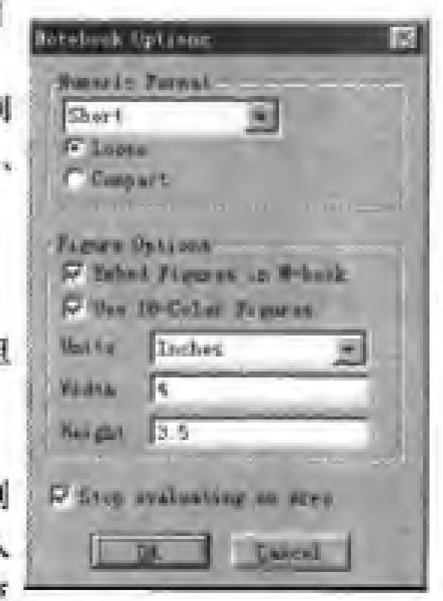


图 8.4　输出格式控制对话框

注意,如果在输入单元中使用 format loose 或 format compact 命令,则控制的将是输出单元与输出单元之间有无空行。

8.4.2 输出图形格式控制

利用如图 8.4 所示的输出格式控制对话框,也可以实现对输出图形的控制。

1.图形镶嵌的控制

决定输入单元中的绘图命令是否向 M - book 文档输出图形的控制选项有两个:一个是输出格式控制对话框中的 Embed Figures in M - book 复选框;另一个是在 Notebook 菜单项中的 Toggle Graph Output for Cell 选项。

M - book 的缺省设置是:Embed Figures in M - book 处于选中状态(即该项的小方块中打✓)。在这种设置下,输出的图形被镶嵌在 M - book 文档中。假如 Embed Figures in M - book 处于不选中状态(即该项的小方块为空白),那么图形将输出到另外的图形窗口中。Embed Figures in M - book 复选框的状态对在其后面运行的所有输入单元中的绘图命令起控制作用。

在 Embed Figures in M - book 复选框被选中的前提下,输出图形是否镶嵌在 M - book 文档中,还取决于 Notebook 菜单项中的 Toggle Graph Output for Cell 选项,该选项相当于一个"拨动开关"。

M - book 的缺省设置是:每个输入单元被定义时,"拨动开关"的拨动次数为 0,此后一定向 M - book 文档输出图形。如果对某输入单元(选中该输入单元,或光标在该输入单元中),选中"拨动开关"奇数次,就会紧跟在该输入单元之后显示"(no graph)"字样。表示以后再执行该输入单元,其输出图形将被抑制。若要使该输入单元恢复输出图形的能力,必须对该输入单元再选中一次"拨动开关",其后"(no graph)"字样也随之消失。

2. 曲面图色彩控制

为了使彩色打印机有正确的彩图输出,也为了使黑白打印机有正确的灰度图形输出,应使输出格式控制对话框中的 Use 16 - Color Figures 复选框处于选中状态(即该项的小方块中打✓)。假如在图形输出前,在输出格式控制对话框中不选择 Use 16 - Color Figures 复选框(即该项的小方块为空白),那么生成的 256 色曲面图就可能得不到正确的色彩表达,更严重的是在黑白打印机上所得到的曲面图将是一片黑色。

3. 图形背景色的控制

在缺省情况下,正常嵌入图形的背景色应是"灰/白"的。假如由于某种原因,所嵌图形出现"灰/黑"背景,那么可以打开输出格式控制对话框,确认 Embed Figures in M - book 处于选中状态,并再次单击 OK 按钮,然后再重新运行输入单元。或在 MATLAB 命令窗口中,运行 whitebg('white')命令后,再重新运行输入单元。

4. 图形大小的控制

在图 8.4 所示的对话框下方有 3 个精确控制图形大小的项目:Units、Width、Height。可通过对这 3 项的设置,决定图形的大小。当然,所得图形也可以像普通 Word 图形一样,对它进行移动、缩放、剪裁和编辑。

8.5 Notebook菜单的其他命令

8.5.1 整个M-book文档输入单元的执行

Notebook菜单项中的Evaluate M-book命令可以把整个M-book文档中的所有输入单元送到MATLAB中去执行。不管光标处于文档的什么位置,执行总是从文档首部开始。在整个M-book文档输入单元执行时,不但会把所有输出单元的内容更新,而且会补写新的输出单元。这个命令在保证整个M-book文档中命令、数据和图形的一致性方面十分有用。但该命令可能会引起版面混乱,所以要慎用。

8.5.2 删去M-book文档中所有输出单元

Notebook菜单项中的Purge Output Cells命令可以删去M-book文档中所有输出单元。具体操作方法为:先选中全部文档,然后选择Notebook菜单项中的Purge Output Cells命令,这样M-book文档中所有输出单元被删去。

8.5.3 单元转化为文本

单元(包括输入单元和输出单元)与文本不同,它们是"活"的。所谓"活"单元,是指当用户改变输入单元并再次运行时,它的输出单元也会更新,即新的结果会覆盖掉原来的结果。假如用户希望保持原来的结果,那么就应该将"活"单元转化为"死"单元(即文本)。

单元转化为文本的方法是:选定单元,再选择Notebook菜单中的Undefine Cells命令。或将光标置于单元之中,按组合键Alt+U。单元转化为文本后其样式会改变,由原来的样式变成Normal样式,因此它的字体、大小、颜色均为Normal样式。

当某输入单元或单元群被转化为文本时,与之相应的输出单元也被自动转化为文本。然而,当输出单元被转化为文本时,就切断了它与输入单元的联系,但输入单元的性质不会因此而改变,此后,若再次运行输入单元,Notebook会紧跟在输入单元之后产生一个新的输出单元,而原来的结果不变。

8.6 M-book模板样式的修改

同其他Word模板一样,用户既可以修改M-book模板原有样式,也可以加入新样式。例如,现有的M-book模板中,输入单元是绿色的,输出单元是蓝色的,错误单元是红色的。现在要把输出单元的颜色变为黑色,操作步骤如下:

(1) 选取Word"格式"菜单中的"样式"选项,弹出样式对话框。

(2) 在"样式"列表框中选择Output,然后单击"更改"按钮,弹出更改样式对话框。

(3) 选中"添至模板"复选框,然后再单击"格式"按钮,并在弹出栏中选"字体",又弹出字体

对话框。

(4) 在字体对话框中,将“颜色”项改为黑色,按“确定”按钮,然后依次退出。

习 题 八

1. 简述 M-book 模板和 Normal 模板的区别。

2. 输入单元的定义和执行有哪些方法? 具体操作如何?

3. 自初始化单元和输入单元有何区别?

4. 自初始化单元和单元群有何作用? 试举例说明。

5. 计算区有何作用? 如何定义和执行?

6. 对数据和图形实施输出控制应选择什么菜单命令?

7. 在 M-book 文档中输入 MATLAB 命令时要注意什么?

8. 利用 MATLAB 笔记本制作一份电子讲稿,要求能现场进行复杂的科学计算或改变参数后进行实时计算,并给出数字或图形结果。在此基础上,进一步尝试利用 MATLAB 笔记本来实施多媒体教学。

第9章 MATLAB环境下的仿真软件Simulink

Simulink是MATLAB环境下对动态系统进行建模、仿真和分析的一个软件包。它于20世纪90年代初由Mathworks公司开发。现在较为流行的版本有：与MATLAB 5.2配用的Simulink 2.2以及与MATLAB 5.3配用的Simulink 3.0。

顾名思义，Simulink的名字表明了该系统的两个主要功能：Simu（仿真）和Link（连接）。即用户可以调用现成的图形模型，并将它们适当地连接起来以构成动态系统的模型，然后对系统进行仿真，并可以随时观察仿真结果和干预仿真过程。Simulink的模块库为用户提供了多种多样的功能模块，这是一笔非常丰富的资源。本章介绍Simulink建模、仿真和分析的基本方法。

9.1 Simulink的基本操作

在安装MATLAB过程中，若选中了Simulink组件，则在MATLAB安装完成后，Simulink也就安装好了。Simulink不能独立运行，只能在MATLAB环境中运行。

9.1.1 Simulink的启动与退出

1. Simulink的启动

启动Simulink的方法有3种：

(1) 在MATLAB的命令窗口直接输入命令simulink。

(2) 单击MATLAB命令窗口工具栏上的Simulink模块库浏览器命令按钮。

(3) 在MATLAB命令窗口File菜单中选择New菜单项下的Model命令。

Simulink启动后会显示如图9.1所示的Simulink模块库浏览器。采用第3种方法同时还会出现如图9.2所示的一个新的模型窗口（名字为untitled）。采用第2种和第3种方法启动Simulink模块库浏览器后再单击其工具栏中的新建模型命令按钮，也会弹出如图9.2的模型窗口。Simulink模块库窗口中显示了各类模块库的图标。用户利用模型窗口，通过鼠标的拖放操作可以创建一个模型。

如果要对一个已经存在的模型文件进行编辑修改，需要打开该模型文件，其也有3种方法：

(1) 在MATLAB命令窗口直接输入模型文件名（不要加扩展名.mdl）。这要求该文件在当前目录下或在已定义的搜索路径中。

(2) 在模型窗口File菜单中选择Open命令，然后选择或输入欲编辑模型的名字。

(3) 单击模型窗口工具栏上的打开命令按钮。

2. Simulink的退出

退出Simulink，只要关闭所有模型窗口和Simulink模块库窗口即可。



图 9.1 Simulink 模块库浏览器

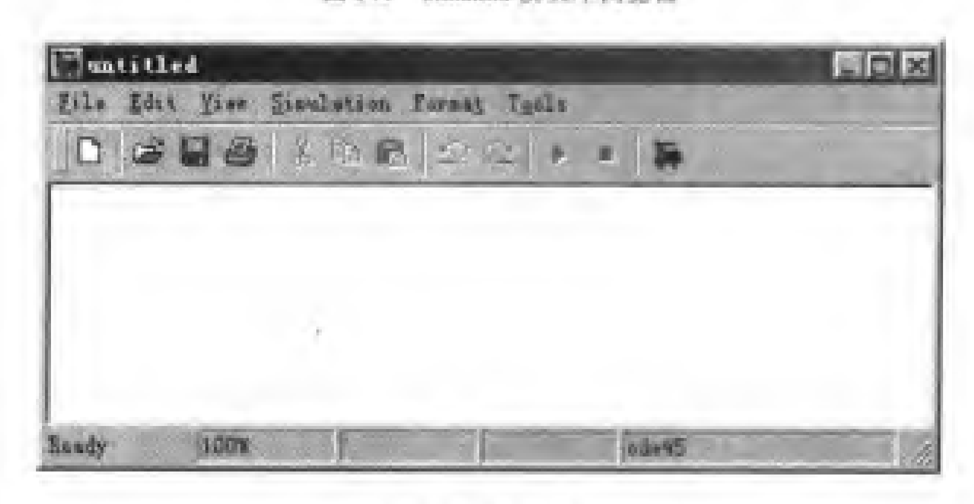


图 9.2 模型窗口

9.1.2 Simulink 模块的操作

模块是建立 Simulink 模型的基本单元。用鼠标将模块拖到模型窗口，再用适当的方式把各种模块连接在一起就能够建立动态系统的模型。Simulink 模块有连续系统(Continuous)、离散系统(Discrete)、非线性系统(Nonlinear)等几类基本系统构成模块，还包括函数与表(Functions & Tables)、数学运算(Math)、信号与系统(Signals & Systems)、输入信号源(Source)以及接收模块(Sinks)。

1. 选取模块

当选取单个模块时，只要用鼠标在模块上单击即可，这时模块的角上出现黑色的小方块。选取多个模块时，在所有模块所占区域的一角按下鼠标左键不放，拖向该区域的对角，在此过程中会出现虚框，当虚框包住了要选的所有模块后，放开鼠标左键，这时在所有被选模块的角上都会

出现小黑块，表示模块都被选中了，如图 9.3 所示。

Sine Wave　Scope

图 9.3　选取多个模块

2. 复制与删除模块

(1) 在不同的窗口之间复制。最简单的办法是用鼠标左键点住要复制的模块（首先打开源模块和目标模块所在的窗口），按住左键移动鼠标到相应窗口（不用按住 Ctrl 键），然后释放，该模块就会被复制过来，而源模块不会被删除。

当然还可以用模块窗口 Edit 菜单中的 Copy 和 Paste 命令或工具栏上的复制和粘贴命令按钮来完成复制。

(2) 在同一模型窗口内复制。有时一个模型需要多个相同的模块，这时的复制方法是：用鼠标左键点住要复制的模块，按住左键移动鼠标，同时按下 Ctrl 键，到适应位置放开鼠标，该模块就被复制到当前位置。更简单的方法是按住鼠标右键（不按 Ctrl 键）移动鼠标。

还可以用模块窗口 Edit 菜单中的 Copy 和 Paste 命令或工具栏上的复制和粘贴命令按钮来完成复制。

模块复制以后，会发现复制出的模块名称在原名称的基础上加了编号，这是 Simulink 的约定：每个模型中的模块和名称是一一对应的，相同的模块或不同的模块都不能用同一名字。

(3) 删除模块。选定模块，选择 Edit 菜单中的 Cut（删除到剪贴板）或 Clear（彻底删除）命令。或者在模块上单击鼠标右键，在弹出菜单上选择 Cut 或 Clear 命令。

3. 模块的参数和属性设置

Simulink 中几乎所有模块的参数都允许用户进行设置。只要双击要设置的模块或在模块上按鼠标右键并在弹出的快捷菜单中选择 Block Parameters 就会弹出参数设置对话框，图 9.4 所示的是正弦波模块的参数设置对话框，用户可以设置它的幅值、频率、相位、采样时间等参数。模块参数还可以用 set _ param 函数设置，这部分内容将在后面介绍。



图 9.4　模块参数设置对话框

每个模块都有一个内容相同的属性设置对话框，如图 9.5 所示。它的打开方式是在模块上按鼠标右键并在弹出的快捷菜单中选择 Block Properties。该对话框包括根据需要设定的 5 个基本属性：



图 9.5　模块属性设置对话框

（1）描述（Description）　是对该模块在模型中用法的注释。

（2）优先级（Priority）　规定该模块在模型中相对于其他模块执行的优先顺序。优先级的数值必须是整数或不输入数值，这时系统自动选取合适的优先级。优先级的数值越小（可以是负整数），优先级越高。

（3）标记（Tag）　用户为模块添加文本格式的标记。

（4）打开函数（Open function）　用户双击该模块时调用的 MATLAB 函数。

（5）属性格式字符串（Attributes format string）　指定在该模块的图标下显示模块的哪个参数，以什么格式显示。一个参数名前是“% < ”，后面跟着“ > ”。属性格式字符串由任意的文本字符串加嵌入式参数名组成，还可以使用换行符“ \ n”，控制在不同的行显示出参数。如果参数的值不是字符串或数字，相应位置会显示 N/S（not supported）。如果参数名无效，该位置将显示“???”。

4．模块外形的调整

（1）改变模块的大小。选定模块，用鼠标左键点住其周围的 4 个黑方块中的任何一个并拖动，这时会出现一个虚线的矩形表示新模块的大小，到需要的位置后释放鼠标即可。

（2）调整模块的方向。选定模块，选择 Format 菜单中的 Rotate Block 命令使模块顺时针方向旋转 90°，选择 Filp Block 命令使模块旋转 180°。显然两次旋转 90°与一次旋转 180°的操作效果是一样的，如图 9.6 所示。

（3）给模块加阴影。选定模块，选择 Format 菜单中的 Show Drop Shadow 命令使模块产生阴影效果。

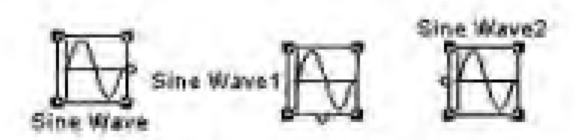


图9.6　调整模块方向

5. 模块名的处理

(1) 模块名的显示。选定模块，选择Format菜单中的Hide Name命令，模块名就会被隐藏，同时Hide Name改为Show Name。选择Show Name命令就会使模块隐藏的名字显示出来。

(2) 修改模块名。用鼠标左键单击模块名的区域，这时会在此处出现编辑状态的光标，在这种状态下能够对模块名随意修改。

模块名和模块图标中的字体也可以更改，方法是选定模块，在Format菜单中选择Font命令，这时会弹出Set Font对话框，在对话框中选择想要的字体。

(3) 改变模块名的位置。模块名的位置有一定的规律，当模块的接口在左右两侧时，模块名只能位于模块的上下两侧，缺省在下侧；当模块的接口在上下两侧时，模块名只能位于模块的左右两侧，缺省在左侧。

因此模块名只能从原位置移动到相对的位置。可以用鼠标拖动模块名到其相对的位置；也可以选定模块，用Format菜单中的Flip Name命令实现相对的移动。

6. 模块的连接

当设置好了各个模块后，还需要把它们按照一定的顺序连接起来才能组成一个完整的系统模型。

(1) 连接两个模块。从一个模块的输出端连到另一个模块的输入端，这是最基本的情况。方法是移动鼠标到输出端，鼠标的箭头会变成十字形光标，这时点住鼠标左键，移动鼠标到另一个模块的输入端，当十字形光标出现"重影"时，释放鼠标左键就完成了连接。

如果两个模块不在同一水平线上，连线是一条折线。要用斜线表示，需要在连接时按住Shift键。两种连线的结果见图9.7。

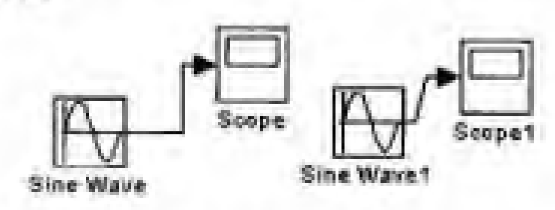


图9.7　两模块不在同一水平线上

(2) 模块间连线的调整。调整模块间连线位置可采用鼠标拖放操作来实现。即先把鼠标移动到需要移动的线段的位置，按住鼠标左键，移动鼠标到目标位置，释放鼠标左键。

还有一种情况，要把一条直线分成斜线段。调整方法和前一种情况类似，不同之处在于按住鼠标左键之前要先按下Shift键，出现小黑方块之后，鼠标点住小黑方块移动，移好后释放Shift键和鼠标。

(3) 在连线之间插入模块。把该模块用鼠标拖到连线上，然后释放鼠标即可。

(4) 连线的分支。实际应用中,经常需要把一个信号输送到不同的模块,这时就需要从一根连线分出一根连线。具体的操作方法是:在先连好一条线之后,把鼠标移到分支点的位置,先按下 Ctrl 键,然后按住鼠标拖动到目标模块的输入端,释放鼠标和 Ctrl 键。

7. 在连线上反映信息

(1) 用粗线表示向量。为了能够比较直观地区别各个模块之间传输的数据是向量还是标量,可以选择模型窗口 Format 菜单中的 Wide Vector Lines 命令,这样传输向量的连线就会变粗。如果再选择 Format 中的 Vector Lines Widths 命令,则在传输向量的连线上方会显示出该向量的长度。

(2) 显示数据类型。在连线上可以显示前一个模块输出的数据类型:选择 Format 菜单中的 Port Data Types 命令。

(3) 信号标记。为了使模型更加直观、可读性更强,可以为传输的信号做标记。建立信号标记的办法是:双击要做标记的线段,出现一个小文本编辑框,在里面输入标记的文本,这样就建立了一个信号标记。信号标记可以随信号的传输在一些模块中进行传递。支持这种传递的模块有 Mux、Demux、From、Selector、Subsystem 和 Enable 等。

要实现信号标记的传递,需要在上面列出的模块的输出端建立一个以"<"开头的标记,当开始仿真或执行 Edit 菜单中的 Update Diagram 命令时,传输过来的信号标记就会显示出来。图 9.8 示意了这个传递的过程。

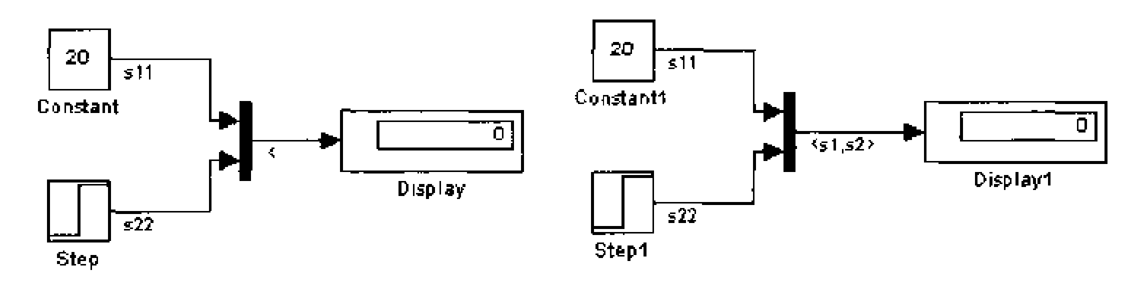


图 9.8 信号标记的传递

9.2 Simulink 的几类基本模块

本节以表格的形式给出了 Simulink 几个基本模块库中的模块功能简介,从而使得读者对 Simulink 的一些主要模块有一个初步的认识。如表 9.1 ~ 9.6 所示,表格中的模块名和模块库中的模块图标下的名称一致。

打开模块库窗口的方法很简单。以连续系统模块库(Continuous)为例,在 Simulink 模块库浏览器窗口中选中 Simulink,然后单击鼠标右键,这时会在鼠标指针的位置弹出一行快捷菜单,内容是 Open the ' Simulink ' Library,用鼠标左键单击该项,出现如图 9.9 所示的 Simulink 基本库窗口,在此窗口中双击 Continuous 模块库的图标即可打开该模块库窗口,如图 9.10 所示。也可以在模块库浏览器窗口中 Simulink 下选中 Continuous 项,然后用与打开 Simulink 基本库窗口同样的方法直接打开连续系统模块库。

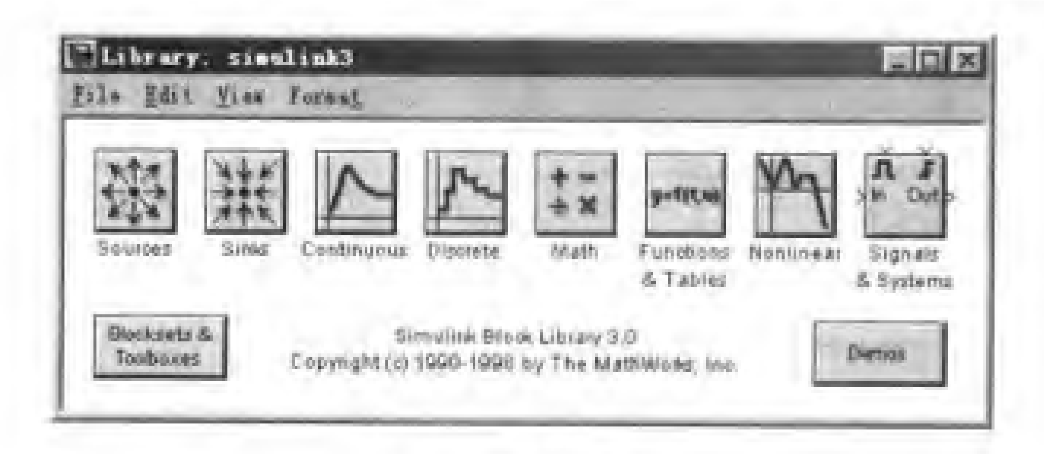


图9.9 Simulink基本库

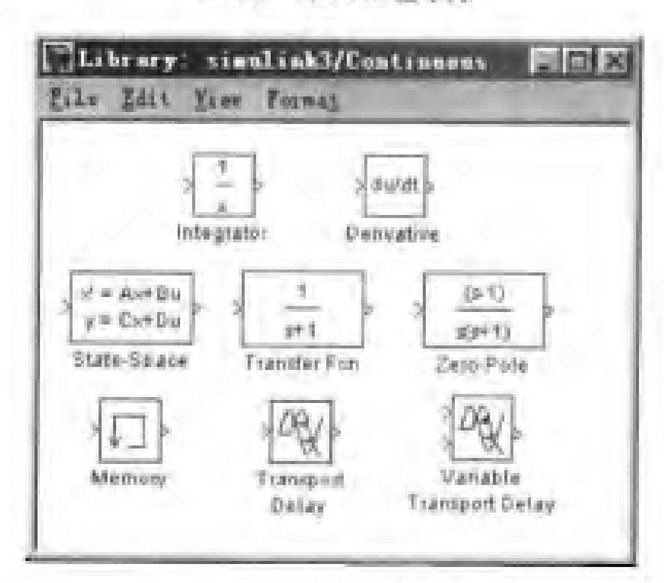


图9.10 连续系统模块库

表9.1 输入源模块(Sources)

模块名	功能简介	模块名	功能简介
Constant	常数	Chirp Signal	频率不断变化的正弦信号
Signal Generator	信号发生器	Clock	输出当前的仿真时间
Step	阶跃信号	Digital Clock	按指定速率输出当前仿真时间
Ramp	线性增加或减小的信号	From File	从文件读数据
Sine Wave	正弦波	From Workspace	从当前工作空间定义的矩阵读数
Repeating Sequence	重复的线性信号,类似锯齿形	Random Number	高斯分布的随机信号
Discrete Pulse Generator	离散脉冲发生器,和采样时间有关	Uniform Random Number	平均分布的随机信号
Pulse Generator	脉冲发生器,和采样时间无关	Band - Limited White Noise	带限白噪声

表 9.2　接收模块(Sinks)

模　块　名	功 能 简 介	模　块　名	功 能 简 介
Scope	示波器	To File	保存到文件
XY Graph	两个信号的关系图,用 MATLAB 图形显示	To Workspace	输出到当前工作空间的变量
Display	实时数值显示	Stop Simulation	输入不为零时停止仿真

表 9.3　连续系统模块(Continuous)

模　块　名	功 能 简 介	模　块　名	功 能 简 介
Integrator	积分环节	Zero – Pole	零极点模型
Derivative	微分环节	Memory	把前一步的输入作为输出
State – Space	状态方程	Transport Delay	把输入信号按给定的时间做延迟
Transfer Fcn	传递函数	Variable Transport Delay	按第二个输入指定的时间把第一个输入做延迟

表 9.4　离散系统模块(Discrete)

模　块　名	功 能 简 介	模　块　名	功 能 简 介
Zero – Order Hold	零阶保持器	Discrete Filter	离散滤波器(IIR、FIR)
Unit Delay	采样保持,延迟一个周期	Discrete Transfer Fcn	离散传递函数
Discrete – Time Integrator	离散时间积分	Discrete Zero – Pole	离散零—极点模型
Discrete State – Space	离散状态方程	First – Order Hole	一阶保持器

表 9.5　信号与系统模块(Signals & Systems)

模　块　名	功 能 简 介	模　块　名	功 能 简 介
In1,Out1	提供输入、输出端口	Data Store Write	写数据到指定的数据存储器
Enable	建立使能子系统	Ground	给未连接的输入端接地,输出 0
Trigger	建立触发子系统	Terminator	连到未连接的输出端,终止输出信号
Mux	把向量或标量组合为大的向量	Data Type Conversion	数据类型转换
Demux	把向量分为标量或小的向量	Function – Call Generator	函数调用发生器
Bus Selector	从输入中选择信号	SubSystem	空的子系统
Selector	选择输入的元素	Configurable Subsystem	表示从用户指定的模块库里选择的任何模块

续表

模　块　名	功 能 简 介	模　块　名	功 能 简 介
Merge	把输入信号合并为输出信号	Model Info	显示模型的修改信息
From	接收标记相同的 Goto 模块的信号	Hit Crossing	检测输入信号的零交叉点
Goto Tag Visibility	定义 Goto 模块标记的有效范围	IC	设置信号的初始值
Goto	把信号送到标记相同的 From 模块	Width	检查输入信号的宽度
Data Store Read	从指定的数据存储器读数据	Probe	检测连线的宽度、采样时间和复数信号标记
Data Store Memory	为数据存储器定义内存区域		

表 9.6　数学运算模块(Math)

模　块　名	功 能 简 介	模　块　名	功 能 简 介
Sum	对输入求代数和	Sign	取输入的正负符号
Product	对输入求积或商	Rounding Function	取整函数
Dot Product	点积(内积)	Combinatorial Logic	建立逻辑真值表
Gain	常量增益(输入乘一个常数)	Logical Operator	逻辑操作符
Slider Gain	可以用滑动条改变的增益	Relational Operator Angle	比较操作符
Matrix Gain	矩阵增益(输入乘一个矩阵)	Complex to Magnitude - Angle	求复数的幅值、相角
Math Function	数学运算函数	Magnitude - Angle Complex	根据幅值、相角得到复数
Trigonometric Function	三角函数	Complex to Real - Imag	求复数的实部、虚部
MinMax	求最值	Real - Imag to Complex	根据实部、虚部得到复数
Abs	求绝对值,或求模(复数)	Algebraic Constraint	强制输入信号为零

9.3　仿真模型参数的设置

前面讨论了对模块的基本操作,但在设计过程中,事先还必须对仿真算法、输出模式等各种模型参数进行设置。设置仿真模型参数的方法有两种:一种是在模型窗口通过菜单命令进行设置。另一种是在 MATLAB 命令窗口中调用有关函数进行设置。本节将分别加以介绍。

9.3.1　通过菜单命令设置仿真模型参数

选择模型窗口 Simulation 菜单中的 Parameter 命令,将出现仿真参数对话框。对话框分 4 个选项卡:Solver(算法)、Workspace I/O(工作空间输入输出)、Diagnostics(诊断)和 Real - Time Workshop

(实时工作间)。

1. Solver 选项卡设置

Slover 选项卡如图 9.11 所示。



图 9.11　Solver 选项卡

(1) 设置仿真起始和停止时间

设置起始和停止时间分别是在 Star time 和 Stop time 两个编辑框内,通过直接输入数值来完成的。时间单位是秒。

(2) 仿真算法的选择

仿真算法根据步长的变化分为固定步长算法和变步长算法。固定步长是指在仿真过程中计算的步长不变,而变步长算法是指在仿真过程中要根据计算的要求改变步长。对于这两种算法,它们所对应的相关选项以及具体算法都有所不同。

在 Type 左栏设定算法类别:变步长(Variable - step)和固定步长(Fixed - step)算法。在 Type 的右栏选择具体算法。

在采用变步长算法时,首先应该指定允许的误差限,包括相对误差限(Relative tolerance)和绝对误差限(Absolute tolerance),当计算过程中的误差超过该误差限时,系统将自动调整步长,步长的大小将决定仿真的精度。在采用变步长算法时还要设置最大步长(Max step size),在缺省值的情况下,系统所给定的最大步长为:

$$最大步长 = (终止时间 - 起始时间)/50$$

在一般情况下,系统所给的最大步长已经足够,但如果用户所进行的仿真时间过长,则缺省值就非常大,有可能出现失真的情况,这时应根据需要设置较小的步长。

在采用固定步长算法时,要先设置固定步长。由于固定步长的步长不变,所以此时不设定误差限,而多了一个模型类型(Mode)的选项,该选项包括:多任务(MultiTasking)、单任务(SingleTask-

ing)和缺省值(Auto)。多任务是指在模型中模块具有不同的采样速率,系统同时检测模块之间采样速率的传递。单任务是指各模块的采样速率相同,不检测采样速率的传递。缺省值则根据模块的采样速率是否相同来决定采用单任务还是多任务。

变步长和固定步长都是算法的模型,这两种模型分别对应了许多种不同的具体算法。在介绍各种算法之前,先介绍一个概念:刚性方程组。所谓刚性,是指问题的解可能在比积分步长还要短的时间间隔内发生很大变化,而人们感兴趣的是在一段相当长的时间内完成的变化。对于一个常微分方程组,如果其 Jacobian 矩阵的特征值相差很大,则称这个方程组为刚性方程组。对于刚性方程组,由于特征值的原因,在计算过程中为了保持稳定性,步长的选取很复杂,所以在解这一类方程组时,必须选取对稳定性要求不是十分严格的算法。在下面的介绍中,将指出哪些算法可以求解刚性问题。

MATLAB 提供的各种变步长算法有:

① discrete (no continuous states) 用于处理非连续状态的模型,过程中不含积分运算。

② ode45(Dormand - Prince) 四、五阶 Runge - Kutta 算法,属于单步法,即每次求解时只需要知道前一步的解而不需要其他的附加条件。通常情况下,该算法对许多问题都是最先使用的方法,所以,MATLAB 都是以 ode45 为缺省算法。该方法只适合求解非刚性问题。

③ ode23(Bogacki - Shampine) 二、三阶 Runge - Kutta 算法,同样属于单步法。该方法也不能求解刚性问题,但对于存在微小刚性的方程组,若允许误差较大,ode23 要比 ode45 效果好。

④ ode113(Adams) 变阶的 Adams 法,是一种多步预报校正算法,每次求解时要用到前几步的解。在对误差范围要求比较严格的情况下,它比 ode45 效果好。该方法不能求解刚性问题。

⑤ ode15s(stiff/NDF) 可变阶的数值微分算法,属于多步法。可以解刚性问题。

⑥ ode23s(stiff/Mod.Rosenbrock) 一种改进的罗森布罗克(Rosenbrock)二阶算法。在对误差要求不高的情况下,效果要比 ode15s 好。可以求解刚性问题。

⑦ ode23t(mod.stiff/Trapezoidal) 一种采用自由内插法实现的梯形公式,适用于无数值衰减的系统。可以求解适当刚性的方程组。

⑧ ode23tb(stiff/TR - BDF2) 一种 TR.BDF2 算法。具体说就是 Runge - Kutta 法的第一级采用梯形公式,第二级采用吉尔(Gear)法,可以解刚性问题。

固定步长的各种算法有:

① discrete(no continue states) 是用于处理非连续状态的模型,过程中不含积分运算。

② ode5(Dormand - Prince) 是一种固定步长的 ode45 算法,属于 Dormand - Prince 算法。

③ ode4(Runge - Kutta) 四阶的 Runge - Kutta 算法。

④ ode3(Bogacki - Shampine) 固定步长的 ode23 算法。

⑤ ode2(Heun) 固定步长的改进欧拉(Euler)算法。

⑥ ode1(Euler) 固定步长的 Euler 算法。

(3) 输出选项设置

对于同样的模型,在输入信号相同的情况下,选择不同的输出方式可以产生不同的效果。MATLAB 提供 3 种输出方式:Refine output、Produce additional output、Produce specified output only。

① Refine output(细化输出) 细化输出的目的是使输出的数据曲线更加平滑。它根据细化系数(Refine factor),可以在每个时间步长内加入若干资料,使得输出资料更加连续、平滑。例如,

细化系数是 3，则把每个步长分成 3 等分，在 1/3 和 2/3 的地方各加入一个资料。细化系数越大，细化后的曲线越平滑。但无限地增大细化系数会导致数据曲线的失真。细化输出的方式比较适合于变步长的算法。

② Produce additional output（产生附加输出）　由用户指定产生输出的附加时刻。选择该项后，右边会产生输出时刻编辑框，在此可以输入附加时刻向量。这种输出方式在仿真计算时改变步长和指定的附加输出时刻相一致。

③ Produce specified output only（仅在指定的时刻产生输出）　仅仅提供在指定的时间点上的输出值。当要比较同一个模型采用不同的仿真方法所得到的结果时，使用这一选项可以产生在一些共同时间点上的输出。

2. Workspace I/O 选项卡设置

Workspace I/O 选项卡如图 9.12 所示。

图 9.12　Workspace I/O 选项卡

(1) 从工作空间中载入数据（Load from workspace）

在仿真过程中，如果模型中有输入端口（In 模块），可从工作空间直接把数据载入到输入端口，即先选中 Input 复选框，再在后面的编辑框内输入数据的变量名。变量名的输入形式有许多种，下面对各种形式分别介绍。

① 矩阵的形式。如果以矩阵的形式输入变量名，则矩阵的列数必须比模型的输入端口数多一个，MATLAB 把矩阵的第一列缺省为时间向量，后面的每一列对应每一个输入端口，矩阵的第一行表示某一时刻各输入端口的输入状态。另外，也可以把矩阵分开来表示，即 MATLAB 缺省的表示方法（t，u），t 是一维时间列向量，表示仿真时间；u 是和 t 长度相等的 n 维列向量（n 表示输入端口的数量），表示状态值。例如，在命令窗口中定义 t 和 u：

```
t=1:10;
u=[t,t.^0.5,t.^2]
```

则 3 个输入端口输入的数据与时间的关系分别为:等于 t、等于 t 的开方和等于 t 的平方,用图 9.13所示的模型的输出可观察各输入端口的输入数据曲线。

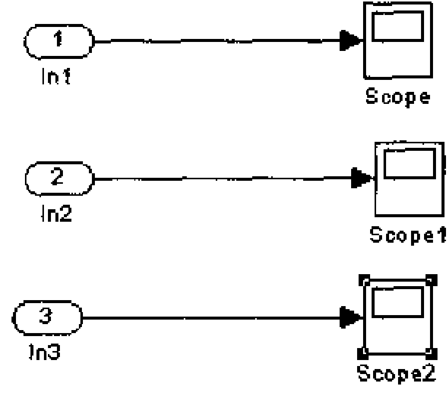


图 9.13　系统模型

② 包含时间数据的结构形式。对于包含时间数据的结构,在 MATLAB 中有非常严格的规定,即在结构中必须有两个名字不能改变的顶级成员:time 和 signals。在 time 成员中包含一个列向量,表示仿真时间;在 signals 成员中包含一个数组,数组中的每个元素对应一个输入端口,并且每个元素必须包含一个名字同样不能改变的 values 成员,values 成员也包含一个列向量,对应于输入端口的输入数据。例如,对于上例,若改为包含数据的结构输入,则命令格式如下:

```
t = 1:10;
A.time = t
A.signals(1).values = t;
A.signals(2).values = t.^0.5;
A.signals(3).values = t.^2;
```

在编辑框内输入 A,则产生的仿真曲线与上面矩阵形式数据输入后的输出曲线完全相同,读者不妨自己试一试。

在一般的结构中,time 成员可以为空,但是必须要有 time 成员。对于上面的例子,若把 time 域置为空,应输入:A.time = [ ],Simulink 根据输入端口的模块参数设置中的采样时间来读取数据,所以,在 In1,In2 等的设置输入端口参数对话框中的采样时间 (Sample Time) 项时必须输入确定的时间值,不能用默认的"-1"来选用动态方式;同时内插数据(Interpolate data)前的复选框不能被选中。

③ 综合形式。在综合形式中,可以把包含时间数据的结构和不包含时间数据的结构分别输入到不同的端口,即为上面两种结构形式的综合。

在 Input 的下面,还有一个 Initial state 的复选框,它表示的是模块的初始化状态。对模块进行初始化的方法是:先选中 Initial state 复选框,然后在后面的编辑框中输入初始化数据的变量名,对于变量要求的几种形式,与前面的输入端口数据的变量形式基本相同,但变量中的数据个数必须和状态模块数相同。

(2) 将输出保存到工作空间(Save to workspace)

在 Save to workspace 区域中,可以选择的选项有:Time(时间)、States(状态)、Output(输出端口)和 Final state(最终状态)。同载入数据的形式一样,保存数据也有矩阵、包含时间数据的结构和综合结构几种形式,在 Save options 区域中的 Format 下可以根据需要进行选择。对于不同的形式来说 Time 的格式是不变的,总是对应仿真的采样时间。下面介绍一下其他几个选项在不同形式下的变化情况。

① 矩阵形式。在使用矩阵的形式时,State 输出的也是矩阵,每一列对应一个模块的状态值,每一行对应一个时刻各个模块的状态值;Output 输出的也是矩阵,矩阵行列的意义与 States 相似,但它只能是输出端口即 Output 的值,如果模型中没有输出端口,则 Output 不能生成矩阵;Final state 同样也是输出矩阵,但输出的矩阵只有一行,表示每个模块的最终状态。

② 包含时间数据的结构形式。同载入数据时介绍的一样，结构包含两个顶级成员：time 和 signals。time 成员为一个列向量，表示仿真时间；signals 成员包括一个子结构数组，子结构的个数应该等于状态模块（States 和 Final state）或输出端口（Output）的个数。每个子结构含有 3 个成员：values、label 和 blockName。values 成员是一个向量，对于 States 而言，它表示状态模块的数据；对于 Final state 而言，它表示状态模块的最终数据；对于 Output 而言，它表示输出端口的数据。label 成员是一个字符串，对于 States 和 Final state，其内容是 Cstate（连续系统模块）或 Dstate _ n（离散系统模块），n 表示相应模块中有 n 组离散状态；对于 Output，其内容是模块间连线上的信号标注。blockName 成员是一个字符串，表示相应模块的名称。

③ 综合形式。和载入数据相似，可以把每个输出端口包含时间数据的结构和不包含时间数据的结构一起输出，而此时 signals 成员中的数组只有一个元素。

（3）保存选项（Save options）

Save options 区中的 Format 选项前面已经介绍过了，而另外两个参数 Limit rows to last 和 Decimation 分别表示限定和选择。Limit rows to last 用来限定保存到工作空间中的数据的最大行数；Decimation 是指从每几个数据中抽取一个，如在编辑框内输入 4，则表示输出数据时，每 4 个数据取一个，也就是每隔 3 个数据取 1 个数据。

3．Diagnostics 选项卡设置

Diagnostics 选项卡如图 9.14 所示。

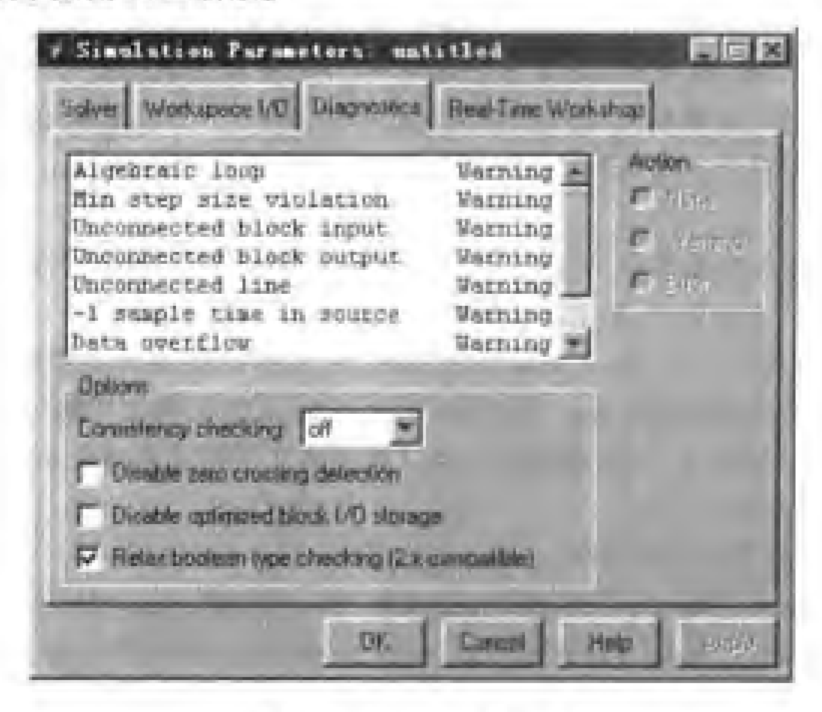


图 9.14　Diagnostics 选项卡

在 Diagnostics 选项卡中，主要是指定系统对一些事件或仿真过程中可能遇到的情况做出什么反应，反应的类型有以下几种：

（1）None　不做任何反应，不影响在任何情况下的程序运行。

（2）Warning　提示警告，但警告信息不影响程序的运行。

(3) Error　提示错误,在提示错误后,运行的程序将停止。

Diagnostics 选项卡中列出了各种可能产生错误的项,用户可以选中任何一项后,在右边的 Action 中根据程序需要进行选择。在对话框下的 Consistency checking 表示一致性检验,如果选择 on,系统会进行一致性检验;如果选择 off,则系统不会进行一致性检验。一致性检验主要是保证模块对一个固定的时间的输出不变,在解决刚性问题中,一致性检验很重要,但由于一致性检验会使系统的性能大大地下降,所以一般情况下,建议把这一项设为 off。

(4) Disable zero crossing detection　表示是否进行零交叉检验。选中时为否,不选时为是。进行零交叉检验会大大提高仿真精度,但速度会相应地下降。

(5) Disable optimized block I/O storage　表示是否进行 I/O 优化存储。当进行优化存储时,会节省大量的内存空间,因此,在进行大型的模型仿真时,最好进行优化存储。

(6) Relax Boolean type checking(2.x compatible)　表示放宽对逻辑数据类型的检查,以与 Simulink 3.0 以下的版本相兼容。当选取这一项时,系统会允许把 double 类型的数据当作逻辑类型进行操作。

4. Real - Time Workshop 选项卡

Real - Time Workshop 作为 Simulink 3.0 的一个重要功能模块,是一种实时开发环境,可以直接从 Simulink 的模型产生可移植的程序源代码(C 语言或 Ada 语言代码)并自动构造出能在多种环境中(包括实时系统和单机仿真)实时执行的程序。通过 Real - Time Workshop 可以在远程处理器上实时运行仿真模型,也可以在主机或外部计算机上运行高速单机仿真。限于篇幅,本书不做深入讨论,读者可参考有关书籍。

9.3.2　在命令窗口设置仿真模型参数

前几节中介绍了使用菜单命令进行仿真和参数设置的方法,可以看出它主要有以下几个优点:

(1) 可以直接设置和修改很多仿真参数,包括仿真时间、算法、仿真步长,等等。

(2) 可以同时对几个系统进行仿真。

(3) 可以实时监视连线上传输的数据。

(4) 可以方便地修改各个模块的参数。

而本节将要介绍的方法是在命令窗口输入函数进行仿真,它虽不及菜单命令简便、直观,但在下述两个方面,它是菜单命令不能替代的:

① 可以仿真 M 文件、MEX 文件(用 C 或 FORTRAN 编写的可以被 MATLAB 调用或执行的动态链接子程序)模型。

② 从 M 文件中直接进行仿真。允许仿真和模块参数反复改变,或者在设定的条件下开始或停止仿真。

从命令窗口运行仿真的函数有 4 个,即:sim、simset、simget 和 set_param。

1. sim 函数

sim 函数的作用是运行一个由 Simulink 建立的模型。其调用格式为:

```
[t, x, y] = sim(modname,timespan,options,data);
```

其中 t 表示仿真的时间向量。x 表示状态模块的状态矩阵。y 表示仿真的输出矩阵,每一列对

应一个输出端口的输出数据。modname 指定模型的名字,用单撇号括起来。timespan 用于指定仿真时间区间,有 3 种格式:[tFinal]指定仿真停止时间,仿真开始时间缺省为 0; [tStart tFinal]指定仿真开始和停止时间;[tStart OutputTimes tFinal]指定开始时间,要输出的时间点和停止时间。options 是由 simset 函数(后面将介绍)设置的仿真参数,数据格式为结构。data 是外部输入到输入端口数据。如果输入到所有端口,则 data 可以是矩阵或结构;如果是输入到单个端口,则 data 只能是结构。

这些参数中只有 modname 是必须的,当其他参数没有设置时,系统会自动以 Simulink Parameters 中设置的参数值为准。

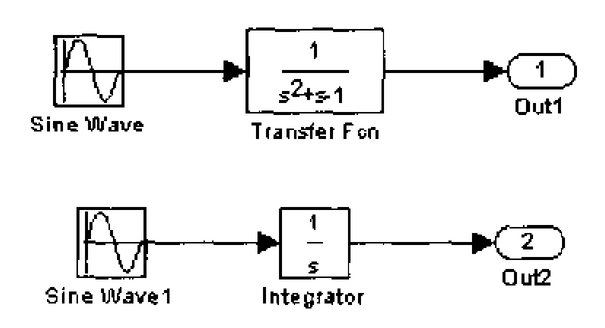


图 9.15 仿真模型 Simul

对于如图 9.15 所示的仿真模型 Simul,我们来比较一下用不同的参数调用 sim 函数的含义。

(1) 在命令窗口输入:

```
[t,x,y] = sim('Simul')
```

这时在 MATLAB 命令窗口用 whos 命令来查看工作空间中的变量,结果如下:

Name	Size	Bytes	Class
t	205x1	1640	double array
x	205x3	4920	double array
xFinal	1x3	24	double array
y	205x2	3280	double array

状态矩阵 x 有 3 列表示模型 Simul 中含有 3 个状态向量,模块 Integrator 含有一个状态向量,而 Transfer Fcn 含有 2 个状态向量,这是因为它是二阶传递函数。而输出矩阵 y 有两列,对应着输出口 Out1 和 Out2 的值。

(2) 指定仿真开始时间、停止时间和中间输出数据的时刻:

```
[t,x,y] = sim('Simul',[2,8])          %时间范围为 2 s ~ 8 s
[t,x,y] = sim('Simul',[2,4,6,8])      %只在 4 个指定的时刻(包括开始和停止时间)有值
```

2. simset 函数

simset 函数用来为 sim 函数建立或编辑仿真参数或规定算法,并把设置结果保存在一个结构变量中。它有如下 4 种用法:

(1) options = simset(property,value,…) 把 property 代表的参数赋值为 value,结果保存在结构 options 中。

(2) options = simset(old_opstruct,property,value,…) 把已有的结构 old_opstruct(由 simset 产生)中的参数 property 重新赋值为 value,结果保存在新结构 options 中。

(3) options = simset(old _ opstruct,new _ opstruct)　用结构 new _ opstruct 的值替代已经存在的结构 old _ opstruct 的值。

(4) simset　显示所有的参数名和它们可能的值。

simset 函数可设置的参数和可能的取值如下：

(1) Solver 属性。该属性指定仿真算法，取值是一个字符串。可以取的值有：ode45、ode23、ode113、ode15s、ode23s 、FixedStepDiscrete、ode5、ode4、ode3、ode2、ode1。

(2) RelTol 属性。表示相对误差容限(Relative error tolerance)，缺省值 1e－3。仅对于变步长算法有用，适用于所有状态变量。

(3) AbsTol 属性。表示绝对误差容限(Absolute error tolerance)，缺省值 1e－6。仅对于变步长算法有用，适用于所有的状态变量。

(4) Refine 属性。表示输出细化因子，缺省值 1。若设置了 Output potions 参数(参数值不是 all)，则此项设置被忽略。

(5) MaxStep 属性。表示最大步长，只对应变步长算法，缺省值 auto。

(6) InitialStep 属性。表示初始步长，只对应变步长算法，缺省值 auto。

(7) MaxOrder 属性。表示 ode15s 算法的最大阶数，只对应算法 ode15s，缺省值 5。

(8) FixedStep 属性。表示固定步长，只对应变定步长算法，如果模型中有离散状态，缺省值为基本采样时间，否则缺省值为仿真时间隔的 1/50。

(9) Outputpoints 属性。表示输出数据点，取值为字符串格式，指定缺省值 specified 时，算法只产生在 timespan 中指定的时间点的值，指定 all 时，输出还会包括算法的时间步长所对应的值。

(10) OutputVariables 属性。表示设置输出变量，取值为字符串格式。如果在设置的字符串中没有 t，x，y，则返回的 t，x 或 y 值为空阵。

(11) SaveFormat 属性。表示设置保存的状态和输出的数据格式，取值为字符串格式，缺省值是 Matrix。

(12) MaxRows 属性。表示限定输出的最大行数，缺省值为 0。

(13) Decimation 属性。设置抽选系数，即在几个数中抽取一个，缺省值为 1。

(14) Initialstate 属性。表示确定连续或离散模块的初始状态值，缺省为空向量。

(15) FinalStateName 属性。表示确定一个变量名用来在仿真结束后保存状态模块的状态值，取值为字符串格式，缺省为空。

(16) Trace 属性。Trace 包括 minstep、siminfo、compile 几个参数。其中参数 minstep 表示在变步长算法的仿真过程中，当某一步不能满足误差限时就停止仿真并提示错误；参数 siminfo 表示在仿真开始时显示一个简略的仿真系数总结；参数 compile 表示对模块的编译过程。

(17) SrcWorkspace 属性。指定在哪个工作空间计算模型中定义的 MATLAB 表达式的值，取值为字符串格式，可能的取值有：base、current、parent。缺省值为 base(基本工作空间，指 MATLAB 命令窗口)。

(18) DstWorkspace 属性。指定在哪个工作空间为模型中定义的变量赋值，取值为字符串格式，可能的取值有：base、current、parent。缺省值为 current(当前工作空间)。

(19) ZeroCross 属性。允许或不允许零交叉检验，零交叉检验只对应于变步长算法，取值为字符串格式，可能的取值有：on、off。缺省值为 on。

例 9.1　以图 9.15 所示的仿真模型为例,说明 simset 函数的用法。

(1) 在命令窗口输入命令:

```
option = simset(' OutputVariables ',' x ',' OutputPoints ',' all ',' FinalstateName ',' date ');
[t,x,y] = sim(' Simul ',[1,10],option)
t = [ ]
x =
        0        0        0
   0.0002   0.0000   0.0002
   0.0012   0.0000   0.0012
   0.0062   0.0000   0.0062
   0.0311   0.0006   0.0317
   0.1520   0.0143   0.1654
   ......   ......   ......
 y = [ ]
```

可见,在 simset 函数中没有出现的参数为空值,而出现的参数无论是缺省值还是设定值都可以显示出来。

(2) 在命令窗口中输入命令:

```
option1 = simset(' OutputVariables ',' xy ',' OutputPoints ',' all ');
[t,x,y] = sim(' Simul ',[2,4,6,8],option1);
t = [2 4 6 8]
x =
              0         0          0
         0.1956    0.6051     0.2375
        -0.0950    0.4792    -1.3763
         1.4455    1.5579    -0.2706
y =
           0          0
      0.6051     0.2375
      0.4792    -1.3763
      1.5579    -0.2706
```

可见,此时表示仿真的时间向量、状态模块的状态矩阵以及仿真的输出矩阵均显示出来。

3. simget 函数

simget 函数用来获得模型的参数设置值。如果参数值是用一个变量名定义的,simget 返回的也是该变量的值而不是变量名。如果该变量在工作空间中不存在(即变量未被赋值),则 simulink 给出一个出错信息。该函数有如下 3 种用法:

(1) struct = simget(modname)　返回指定模型的仿真参数设置(结构)。

(2) value = simget(modname,property)　返回指定模型参数 property 的值。

(3) value = simget(Option Structure,property)　获取结构 OptionStructure 中参数 property 的值如果在该结构中未指定该参数,则返回一个空阵。

用户只需输入能够惟一识别它的那个参数名称的前几个字符即可。对参数名称中字母的大

小写不做区别。

4. set _ param 函数

set _ param 函数的功能很多,这里只介绍如何用 set _ param 函数设置 Simulink 仿真参数以及如何开始、暂停、终止仿真进程或者更新显示一个方块图。

(1) 设置仿真参数

调用格式为:

```
set _ param(modname,property,value,…)
```

其中 *modname* 为设置的模型名,*property* 为要设置的参数,*value* 是设置值。这里设置的参数可以有很多种,而且和用 simset 设置的内容不尽相同,相关参数的设置可以参考有关资料。

(2) 控制仿真进程

调用格式为:

```
set _ param(modname,' SimulationCommand ',' cmd ')
```

其中 *modname* 为仿真模型名称,而 *cmd* 是控制仿真进程的各个命令,包括 start、stop、pause、continue 或 update。

在使用函数的时候,需要注意必须先把模型打开。仍以前面提到的模型 Simul 为例,首先把模型打开,然后调用如下函数:

```
set _ param(' Simul ',' StartTime ',' 10 ',' StopTime ',' 20 ')
set _ param(' Simul ',' SimulationCommand ',' start ')
```

这时模型就会开始仿真。

这一节中介绍了系统仿真的两种方式:菜单命令的方式简便直观,易于使用,但不够灵活;从命令窗口输入函数的方式比较复杂,但它可以很灵活地用于 M 文件或 MEX 文件中,这样就可以编制结构比较复杂的程序,仿真过程可以在各种流程中调用。

下面介绍连续系统建模及仿真应用实例。

例 9.2　假设从实际应用领域(力学、电学、生态或社会)中,抽象出有初始状态为 0 的二阶微分方程 $x''+0.2x'+0.4x=0.2u(t)$, $u(t)$是单位阶跃函数。用积分器直接构造求解微分方程的模型 exm1.mdl 并仿真。

步骤如下:

(1) 改写微分方程。把原方程改写为

$x''=0.2u(t)-0.2x'-0.4x$

(2) 利用 Simulink 模块库中的标准模块构建模型。利用积分模块构造微分方程求解模型的核心思想是:x''经积分作用得x',x'再经积分模块作用就得x,而x'和x经代数运算又产生x''。构建的模型如图 9.16 所示。

在构建模型过程中,对各模型块进行了如下配置:

① u(t)输入模块　它的 step time 被设置为 0,模块名称由原来的 step 改为 u(t)。

② Gs 增益模块　增益参数 Gain 设置为 0.2。

③ 求和模块　其图标形状 Icon shape 选择 rectangular,使模块呈矩形。符号列表 List of signs 设置为 + － －。

④ 积分模块　只是把它们的名称分别改变为 Int1, Int2。

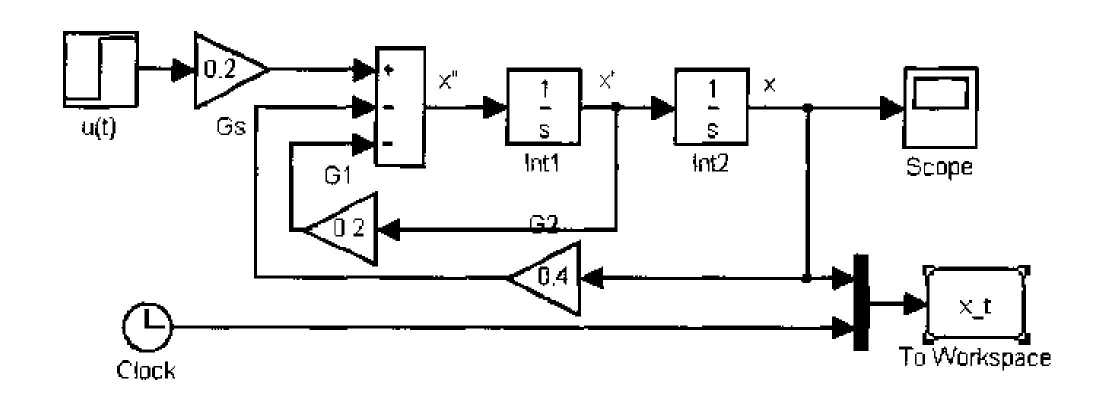


图 9.16 求解微分方程的 Simulink 模型

⑤ G1 和 G2 增益模块 它们的方向旋转可借助 Format 菜单中的 Rotate Block 命令实现。

⑥ Scope 示波器 先双击该模块，出现示波窗；单击示波窗第 6 个工具按钮，引出属数设置界面；在 Data history 选项卡中，选中 Save data to workspace。这将使送入示波器的数据同时被保存在 MATLAB 基本空间的缺省名为 ScopeData 的结构矩阵中。

⑦ Clock 模块 产生仿真时间数据，仅供 To workspace 模块用。

⑧ Mux 模块 模型中的位移数据 x 与时间数据 t 组合成向量。

⑨ To workspace 模块 专为演示而设置。位移数据 x 与时间数据 t 将以 $x-t$ 名称存放在 MATLAB 工作空间。

⑩ 模型窗 exm1.mdl 选中 Simulation 菜单中的 Parameters 命令，打开仿真参数设置窗口。在 Solver 选项卡中，把仿真的停止时间 Stop time 设置为 20。又为演示需要，在 Workspace I/O 选项卡中，选中 Time 和 States 栏，使模型仿真中产生的时间数据以 tout，状态以 xout 名称保存在 MATLAB 工作空间。

(3) 仿真操作。双击示波器图标，打开示波窗。选择模型窗中 Simulation 菜单中的 Start 命令，就可在示波窗中看到位移 x 的变化曲线。

(4) 保存在 MATLAB 工作空间中的数据。为演示起见，同时采用如下 3 种独立途径向 MATLAB 基本空间存放仿真数据：

① 示波器模块向工作空间存放结构数组 ScopeData。

② To workspace 模块以选定的矩阵方式向工作空间存放数组 $x-t$。

③ 模型窗的 I/O 选项卡，以 tout、xout 名称存放数据在工作空间。

注意：用户在实际使用时，可据具体环境，采用以上任何一种方法保存仿真数据。换句话说，以上 3 组数据中的任何一组都可独立地供用户进一步分析时使用。

例 9.3 利用传递函数模块建模。

不失一般性，仍以下面的二阶微分方程为例。假设初始状态为 0，$u(t)$是单位阶跃输入。

$$x''+0.2x'+0.4x=0.2u(t)$$

对方程两边施行拉普拉斯(Laplace)变换，得

$$s^2X(s)+0.2SX(s)+0.4X(s)=0.2U(s)$$

经整理得传递函数

$$G(s)=\frac{X(s)}{U(s)}=\frac{0.2}{s^2+0.2s+0.4}$$

于是根据此式，就可以用Simulink库中的传递函数模块构建求解微分方程的模型。步骤如下：

(1) 根据系统传递函数构造如图9.17所示的模型exm2.mdl。

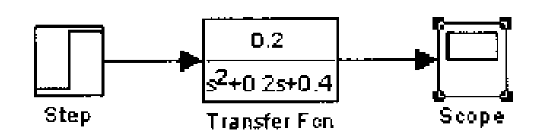


图9.17　由传递函数模块构成的仿真模型

在模型构造过程中，对各模块进行了如下操作：

① U(S)模块　设置Step time为0。

② G(S)模块　双击标准模块，引出对话框；在分子、分母栏中填写所需的系数。

③ 在exm2模型窗中，选Simulation菜单中的Parameters命令，打开仿真参数设置窗。在Solver选项卡中，把仿真的停止时间Stop time设置为20。在Workspace I/O选项卡中，把初始状态设置为(0;0)。

(2) 仿真操作。双击Scope模块，打开示波窗，再选中Simulation菜单中的Start命令，就可看到位移波形。

例9.4　利用状态方程模块建模。

仍以二阶微分方程 $x''+0.2x'+0.4x=0.2u(t)$ 为例。

若令 $x_1=x, x_2=x'$，那么微分方程 $x''+0.2x'+0.4x=0.2u(t)$ 可写成

$$\dot{\boldsymbol{x}}=\begin{vmatrix}\dot{x}_1\\ \dot{x}_2\end{vmatrix}=\begin{vmatrix}0 & 1\\ -0.4 & -0.2\end{vmatrix}\quad\begin{vmatrix}x_1\\ x_2\end{vmatrix}+\begin{vmatrix}0\\ 0.2\end{vmatrix}u(t)$$

写成状态方程为

$$\begin{cases}\dot{\boldsymbol{x}}=\boldsymbol{Ax}+\boldsymbol{Bu}\\ \boldsymbol{y}=\boldsymbol{Cx}+\boldsymbol{Du}\end{cases}$$

式中 $\boldsymbol{A}=\begin{vmatrix}0 & 1\\ -0.4 & -0.2\end{vmatrix}$，$\boldsymbol{B}=\begin{vmatrix}0\\ 0.2\end{vmatrix}$，$\boldsymbol{C}=|1\quad 0|$，$\boldsymbol{D}=\boldsymbol{0}$。$u$ 可以是任意形式的输入。

在Simulink库中有标准的状态方程模块可供调用。假设输入的函数 u 是单位脉冲函数 $\delta(t)$，研究该系统的位移变化。

(1) 单位脉冲函数的数学含义及近似实现。单位脉冲函数在数学上定义为

$$\begin{cases}\delta(t-a)=0, & t\neq a\\ \int_{-\infty}^{\infty}\delta(t)\mathrm{d}t=1\end{cases}$$

像其他物理体系中不存在理想单位脉冲一样，Simulimk库中也没有现成的单位脉冲标准模块。为此，必须采用某种近似方法产生。单位脉冲近似构建的思路是，用一个面积为1的“窄高”脉冲近似。近似脉冲宽度的选择要考虑两方面的因素：脉冲宽度应远小于被研究系统的最快动态模式(如特征根的实部)。脉冲宽度不能太小，以免引起严重的圆整或截断误差。由于本系统特征根的实部绝对值为0.1，所以取脉冲宽度为0.01，幅度为100。即 $\delta(t)=100u(t)-100u(t-0.01)$。

(2) 利用库模块构造如图9.18所示的仿真模型exm3.mdl。

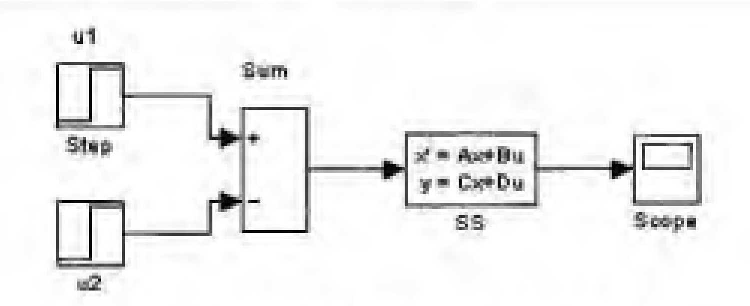


图 9.18　用状态方程模块构成的仿真模型

在构建模型时，对各模块进行了如下操作：

① u1 模块。它的阶跃时间 Step time 设置为 0，(最)终(幅)值为 100。

② u2 模块。它的阶跃时间 Step time 设置为 0.01，(最)终(幅)值为 100。

③ Sum 求和模块。符号列表填写 + －。

④ SS 状态方程即 state space 模块。A，B，C，D 栏依次填定[0,1；-0.4，-0.2]，[0,0.2]，[1，0]，0。

选中模型窗口 Simulation 菜单中的 Parameters 命令，打开仿真参数设置窗口。在 Solver 选项卡中，把仿真的停止时间 Stop time 设置为 20。

(3) 执行仿真操作，仿真结果见图 9.19。



图 9.19　仿真结果

9.4　子系统的建立与封装

当模型较大或者较复杂时，可以把几个模块组合成一个新的模块，并且新模块能够完成几个模块的功能，那么这样的模块称为子系统。子系统的优点是：减少系统中的模块数目，使系统易于调试，而且建立的子系统具有完整的功能，可以在其他模型中直接作为模块使用。

9.4.1　子系统的建立

建立子系统有两种方法：通过 Subsystem 模块建立子系统和通过已有的模块建立子系统。两者的区别是：前者先建立子系统，再为其添加功能模块；后者先选择模块，再建立子系统。下面就

这两种方法分别进行讨论。

1. 通过 Subsystem 模块建立子系统

操作步骤为：

(1) 先打开 Simulink 模块库浏览器，新建一个仿真模型。

(2) 打开 Simulink 模块库中的 Signals & Systems 模块库，复制 Subsystem 模块到新的模型窗口中。

(3) 用鼠标左键双击 Subsystem 模块打开一个空白的 Subsystem 窗口，将要组合的模块添加到该窗口中，另外还要根据需要复制一个或多个 Input 和 Output 模块，表示子系统的输入和输出端口。这样，一个子系统就建好了。

例如，将积分模块和微分模块组合建立一个子系统，子系统的模型和子系统模块的图标分别如图 9.20 和图 9.21 所示。

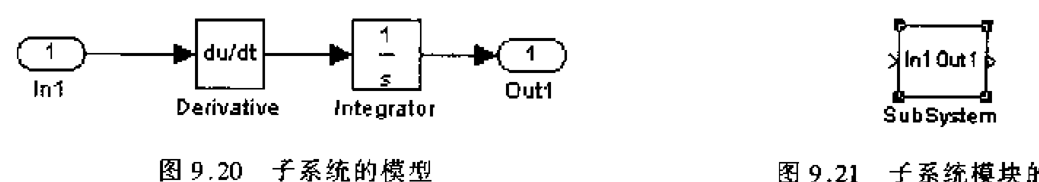


图 9.20　子系统的模型　　　图 9.21　子系统模块的图标

2. 通过已有的模块建立子系统

操作步骤为：

(1) 先选择要建立子系统的模块，不包括输入端口和输出端口。

(2) 选择模型窗口 Edit 菜单中的 Create Subsystem 命令，这样，子系统就建好了。在这种情况下，系统会自动把 Input 模块和 Output 模块添加到子系统中，并把原来的模块变为子系统的图标。

9.4.2　子系统的条件执行

子系统的执行可以由输入信号来控制，用于控制子系统执行的信号称为控制信号，而由控制信号控制的子系统称为条件执行子系统。在一个复杂模型中，有的模块的执行依赖于其他模块，这种情况下，条件执行子系统是很有用的。条件执行子系统分为：使能子系统、触发子系统和使能加触发子系统。

1. 使能子系统

使能子系统表示子系统在由控制信号控制时，控制信号由负变正时子系统开始执行，直到控制信号再次变为负时结束。控制信号可以是标量也可以是向量。如果控制信号是标量，则当标量的值大于 0 时子系统开始执行。如果控制信号是向量，则向量中任何一个元素大于 0，子系统将执行。

建立使能子系统的方法是：打开 Simulink 模块库中的 Signals & Systems 模块库，将 Enable 模块复制到子系统模型中(如图 9.22 所示)，则系统的图标发生了变化(如图 9.23 所示)。

下面通过一个实例说明使能子系统的工作原理。

例 9.5　利用使能子系统构成一个正弦半波整流器。

操作步骤如下：

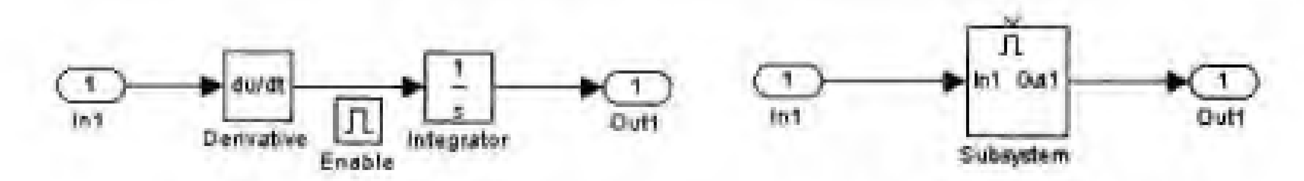


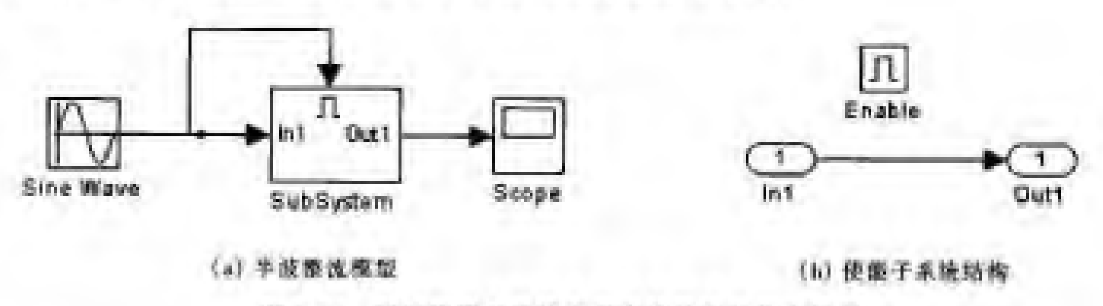
图 9.22 使能子系统　　　　图 9.23 使能子系统的模块图标

(1) 打开 Simulink 模块库浏览器以及新建一个模型窗口。将模块库浏览器中的 Sinewave、Subsystem 和 Scope 3 个模块拖至新建的模型窗口。

(2) 双击空子系统模块 Subsystem,打开其结构模型窗。

(3) 将 Simulink 浏览库中的 In 模块、Out 模块、Enable 模块拖至子系统的结构模型窗。把 In 模块的输出直接送到 Out 模块的输入端,Enable 模块无须进行任何连接,且采用它的缺省设置实现使能子系统。

(4) 按图 9.24 完成新建模型窗口中各模块间的连接并存盘。



(a) 半波整流模型　　　　(b) 使能子系统结构

图 9.24 利用使能子系统实现半波整流的仿真模型

(5) 双击示波器模块,打开显示窗。然后选择 Simulink 菜单中的 Start 命令,就可看到如图 9.25所示的半波整流波形。

说明:

(1) 使能子系统外观上有一个"使能"控制信号输入口。"使能"是指:当且仅当"使能"输入信号为正时,该模块才接收 In 输入端的信号。

(2) 使能子系统建立好后,要对 Enable 模块进行参数设置。双击 Enable 模块打开参数对话框。选中复选框 Show output port 后,可以为 Enable 模块添加一个输出端,用以输出控制信号。在 States when enabling 选项列表框中有两个选项:held 和 reset。Held 表示当使能子系统停止输出后,输出端口的值保持最近的输出值;reset 表示当使能子系统停止输出后,输出端口重新设为初始值,在此选择 reset。对设置完的系统模型即可进行仿真。

图 9.25 半波整流仿真波形

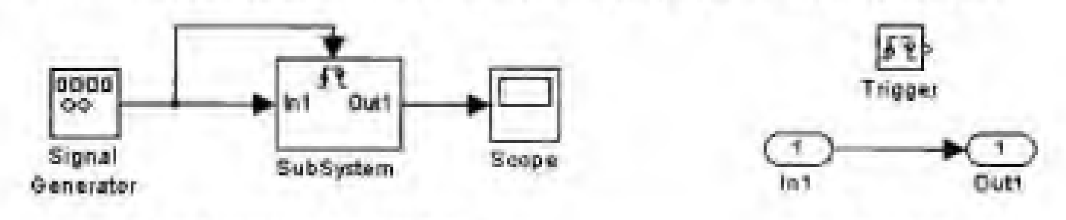
2. 触发子系统

触发子系统是指当触发事件发生时开始执行子系统。与使能子系统相类似,触发子系统的建立要把 Signals & Systems 模块库中的 Trigger 模块拷贝到子系统中。

例 9.6 利用触发子系统将一锯齿波转换成方波。

操作步骤如下:

(1) 仿上例步骤,用 Signal Generator、Subsystem 和 Scope 模块构成如图 9.26 所示的子系统,双击 Signal Generator 模块图标在 Wave from 的下拉列表框中选择 sawtooth,即锯齿波。



(a) 将锯齿波转换为方波的模型　　(b) 触发子系统结构

图 9.26　利用触发子系统将锯齿波转换为方波

(2) 将上例中的 Enable 模块换成 Trigger 触发模块。双击该模块并选 Trigger type 触发事件形式为 either,即上跳沿或下跳沿触发。

(3) 按图 9.26 完成新建模型窗口中各模块间的连接。并存盘。

(4) 双击示波器模块,然后选择 Simulink 菜单中的 start 命令,就可看到图 9.27 所示的方波。



图 9.27　将锯齿波转换为方波的仿真结果

说明:

(1) 触发子系统外观上有一个"触发"控制信号输入口。子系统一旦被触发,在每次触发结束到下次触发之前总是保持上一次的输出值,而不会重新设置初始输出值。

(2) 触发形式由子系统内的触发模块对话框中 Trigger type 的下拉列表框选择:

rising(上跳沿触发)　控制信号从负值或 0 上升到正值时子系统开始执行。

falling(下跳沿触发)　控制信号从正值或 0 下降到负值时子系统开始执行。

either(上跳沿或下跳沿触发)　当控制信号满足上跳沿或下跳沿触发条件时,子系统开始执行。

funtion - call(函数调用触发)　表示子系统的触发由 S 函数的内部逻辑决定,这种触发方式必须与 S 函数配用。

(3) 双击 Trigger 模块后,还有一复选框 Show output,表示是否为 Trigger 模块添加一个输出端,选中后还可以选输出信号的类型。

3. 使能加触发子系统

所谓使能加触发子系统就是把 Enable 和 Tirgger 模块都加到子系统中,使能控制信号和触发控制信号共同作用子系统的执行,也就是前两种子系统的综合。该系统的行为方式与触发子系统相似,但只有当使能信号为正时,触发事件才起作用。

9.4.3 子系统的封装

在对子系统进行参数设置时，需要打开其中的每个模块，然后分别进行参数设置，子系统本身没有基于整体的独立操作界面，从而使子系统的应用受到很大的限制，为解决这些问题，Simulink 提供了子系统封装技术。

所谓子系统的封装，就是为子系统定制对话框和图标，使子系统本身有一个独立的操作界面，把子系统中的各模块的对话框合成一个对话框，在使用时不必打开每个模块进行参数设置，这样使子系统的使用更加方便。

子系统的封装过程很简单：先选中所要封装的子系统，再选择模型窗口 Edit 菜单中的 Mask Subsystem 命令，这时将出现 Mask Editor 对话框（如图 9.28 所示）。



图 9.28 Mask Editor 对话框

Mask Editor 对话框中共包括 3 个选项卡：Icon、Initialization 和 Documentation。子系统的封装主要就是对这 3 项参数进行设置，下面针对这 3 个选项卡来介绍子系统的封装。

1. Icon 选项卡的参数设置

在 Icon 选项卡中主要设置封装模块的图标。Icon 选项卡中包括两个编辑框：Mask type 和 Drawing commands，以及几个设置封装图标特性的弹出式列表框（如图 9.28）。其中 Mask type 编辑框在 3 个选项卡中都有，它表示封装的类型，而这种类型只有表示文字的意义，可以输入任何字符串，输入的字符串将显示在封装模块对话框的顶部。下面将着重介绍 Drawing commands 编

辑框和弹出式列表框的设置。

(1) Drawing commands 编辑框

该编辑框主要用来建立封装图标,并且可以在封装图标中显示文本、图形、图像或传递函数。

在封装图标中显示文本的函数有 4 个:disp、text、fprintf、port _ lable。下面介绍 port _ lable 函数的用法。

port _ lable 函数根据端口类型和端口号为端口添加标记。其调用格式为:

```
port_lable('port_type', port_num,'lable');
```

以图 9.29 的触发子系统为例,在 Drawing commands 编辑框中输入如下命令:

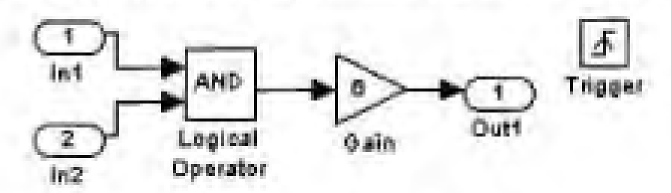


图 9.29　触发子系统

```
disp('LOGIC');
port_label('input', 1,'In1');
port_label('input', 2,'In2');
port_label('output', 1,'Out1');
```

则新生成的子系统图标如图 9.30 所示。

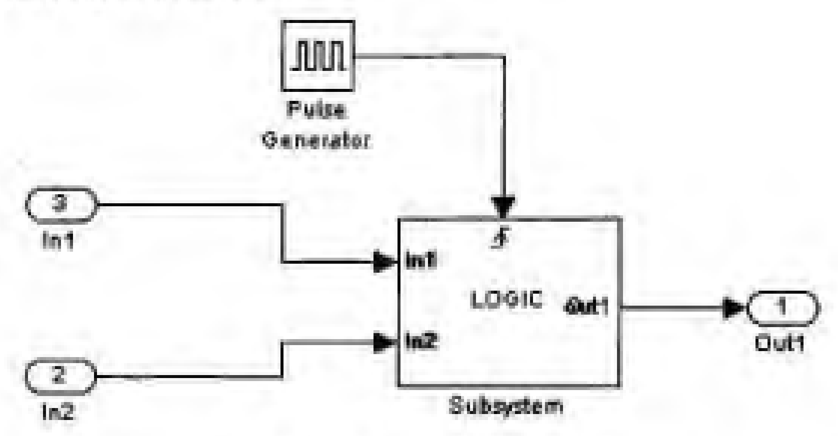


图 9.30　显示文本的子系统图标

在图标中显示图形可以用 plot 函数。在图标中显示图像可以用 image 函数。在图标中显示传递函数使用 dpoly 函数,其调用格式为:

① dpoly(A,B)

其中 A 为传递函数的分子向量,B 为传递函数的分母向量。

② dpoly(A,B, 'character')

当显示的传递函数按 x 的降幂排列时,*character* 的取值为 x,按 $1/x$ 升幂时取 $x-$。

③ droots(z,p,k)

表示显示零极点模型的传递函数，其中 z 为零点，p 为极点，k 为增益。

例如，输入如下命令：

dpoly([1,2],[2,3,4],'x')

生成的子系统的图标如图 9.31 所示。

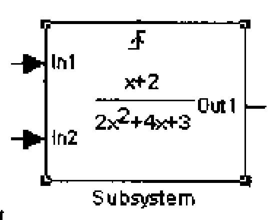


图 9.31　显示传递函数的子系统图标

(2) 设置封装图标特性

在 Drawing commands 编辑框下面的下拉式列表框中，可以分别对图标的各种特性进行设置。

① Icon frame　设置图标外的边框。Invisible 表示将边框隐藏，Visible 表示显示边框。

② Icon transparency　设置图标的透明度。Transparent 表示透明，显示图标中的内容。Opaque 表示不透明，不显示图标中的内容。

③ Icon rotation　设置图标是否跟模块一起翻转，Fixed 表示不翻转，Rotates 表示翻转。

④ Drawing coordinates　设置在 Drawing commands 编辑框中使用命令 plot 和 text 时的坐标系。Pixel 表示图标以像素为单位，当模块大小改变时，图标不随之改变；Autoscale 表示规定图标的左下角的坐标为(0,0)，右上角的坐标为(1,1)，要显示的文本等必须把坐标设在[0,1]之间才能显示。当模块大小改变时，图标也随之改变；Normalized 表示根据设定的坐标点自动选取坐标系，使设置中的最小坐标位于图标左下角，最大坐标位于图标右上角。当模块大小改变时，图标也随之改变。

2. Initialization 选项卡的参数设置

Initialization 选项卡的参数设置对话框如图 9.32 所示。其中 Mask type 编辑框与前面介绍的内容一样，在此不再赘述。本节主要介绍设置参数的提示符、变量名和初始化命令。

(1) 提示符和变量名

Initialization 选项卡中间的 Variable 和 Prompt 编辑框分别用来指定用户需要设置的变量名和它们的提示符。它们的作用是在封装子系统的参数对话框中提示用户设置什么内容和指定变量来接受用户设置的内容。例如，要在子系统中设置两个变量：输入 1(in1)和输入 2(in2)，则设置的步骤为：

① 在 Prompt 后的编辑框中输入提示符“输入 1”，在 Variable 后的编辑框中输入变量名 in1。

② 用鼠标左键单击对话框左上方的 Add 按钮，则提示符“输入 1”和变量名 in1 都被加入到列表中，同时编辑框 Prompt 和 Variable 变为空，此时可以继续输入其他提示符和变量名。

③ 待所有提示符和变量名都设置好了，单击参数设置对话框下面的 OK 按钮，再用鼠标左键双击子系统模块，则出现设置好的子系统参数设置对话框。

注意：若需要改变提示符和变量名在封装后的子系统对话模型中显示的顺序，可以选中该行后按下 Up 或 Down 按钮进行调节。

另外，参数 Assignment 的设置也与变量有关。其选项有 Evaluate 和 Literal 两项：Evaluate 表示用户输入的内容先由 MATLAB 进行计算，再把结果赋给相关变量；Literal 表示用户输入的内容不经过计算，直接把字符串的格式赋给相关变量。

(2) 控制类型

控制类型是指控制封装后的子系统参数设置对话模型所提供的设置参数的方式。从图9.32



图9.32 Initialization选项卡

中Control type下拉式列表框可知控制类型有：

① Edit 选择该项时，为用户提供一个编辑框，通过编辑框可以设置参数的值或表达式。

② Checkbox 选择该项时，只提供一个复选框，选中与不选中复选框会返回不同的值。如果参数Assignment设为Evaluate，那么选中复选框时，返回值为1，不选中时，返回值为0；如果参数Assignment设为Literal，那么选中复选框时，返回值为on；不选中时，返回值为off。

③ Popup 选择该项时，为用户提供一个弹出式菜单，并且Popup string编辑框会被激活，在此输入弹出式菜单的各选项，选项之间用符号"|"分开。

如果参数Assignment改为Evaluate，那么选中菜单中的某个选项时，返回值是该选项所在菜单中的序号；如果参数Assignment改为Literal，那么选中菜单中的某选项时，以该选项的字符串形式作为返回值。

(3) 设置初始化命令

初始化命令的设置在对话框下面的Initialization commands编辑框内进行，在此输入初始化命令，而这些初始化命令将在仿真开始、载入模型、更换模块图标和重设封装子系统的图标时被调用。初始化命令主要用来定义封装子系统的变量，这些变量可以被所有封装定义的初始化命令、封装子系统中的模块和绘制图标的命令所使用。

初始化的命令由MATLAB中的表达式组成，其中包括：MATLAB函数、操作符和封装子系统工作区中定义的变量，但这些变量不包括基本工作区中的变量。

对于封装工作区定义的变量，通过初始化命令和模块的参数相联系，也就是说模块的参数在获取数据时，先读取封装工作区的变量值，再通过初始化命令来取值。

3．Documentation 选项卡的参数设置

在 Documentation 选项卡中(如图 9.33 所示)所要设置的内容除了前面已经介绍过的封装类型外，还有描述文本和帮助文本。



图 9.33　Documentation 选项卡

(1) Block description 编辑框

Block description 编辑框中设置的内容将显示在封装模块对话框的上部，文本的内容一般是对模块功能的描述，用户也可以根据自己的需要自行设置。

(2) Block help 编辑框

Block help 编辑框输入的内容为帮助文件，当按下封装子系统参数对话框的 Help 按钮时，将显示输入的内容。

9.5　在命令窗口中创建模型

这一节将介绍构造仿真系统模型、设置模块参数和模型参数的另一种操作方式：使用命令编写程序。这种方式虽然不像构造 Simulink 框图那样简便、直观，但它可以灵活地控制程序的流

程，并且参数可以动态设置和修改，这种方式可以实现很多框图方式无法实现的功能。

9.5.1 构造模型的命令

在表 9.7 中列出了这些命令和它们的功能。命令的具体用法这里不再一一介绍，使用时请参考在线帮助。

表 9.7 构造模型的命令列表

命　　令	功　　能	命　　令	功　　能
add _ block	在系统中加入一个新模块	gcbh	获取当前模块的句柄
add _ line	在系统中加入一条线	gcs	获取当前系统的路径名
bdclose	关闭一个系统窗口	get _ param	获取参数值
bdroot	获取根系统的名字	new _ system	建立一个新的 Simulink 系统
close _ system	关闭一个系统窗口	open _ system	打开一个存在的系统
delete _ block	从系统中删除模块	replace _ block	替换系统中的模块
delete _ line	从系统中删除连线	save _ system	保存系统
find _ system	查找系统、模块、连线或注释	set _ param	设置参数值
gcb	获取当前模块的路径名	simulink	打开仿真模块库

有很多命令需要指定 Simulink 系统或模块，这需要同时指定它们的位置。有以下几种情况：

(1) 指定一个系统　要指定包含该系统的文件名和文件的路径。

(2) 指定一个子系统　要指定该子系统所处的系统和其上的各级子系统。假定需要指定的子系统为 subsystem，那么格式为：

```
system/subsystem1/ subsystem2/…/ subsystem
```

(3) 指定一个模块　需要指定模块所处的系统和各级子系统。假定要指定的模块名为 block，那么格式为：

```
system/subsystem1/ subsystem2/…/ subsystem/block
```

如果模块名不止一行，那么需要加入 sprintf('\n')作为分行的标志。例如，如果要指定的模块是 Pulse Generator(脉冲发生器)，那么如下命令可以获取模型 MyMode 中的 Pulse Generator 模块的参数 Period 的值。

```
get _ param(['MyModel/ Pulse',sprintf ('\n'),'Generator'], 'Period')
```

有的模块名中含有斜线“/”，要指定该模块名时，需要把斜线重复一次。

9.5.2 设置模块参数

set _ param 函数用来设置仿真模型或模块的各种参数。模型参数用来设置模型的仿真参数、打印选项、回调(callback)参数等内容。模型的回调参数设置在什么时候执行指定的回调程序。

前面曾介绍过利用 set _ param 函数来设置仿真参数。下面介绍如何利用该函数来设置各种模块参数，包括模块的通用参数、模块的特定参数、模块的回调(callback)参数。下面的命令设置

仿真的开始时间为 10 s，终止时间为 100 s：

```
set _ param(' MyModel ',' StarTime ',10,' StopTime ',100)
```

下面命令设置开始仿真时执行命令 notebook：

```
set _ param(' MyModel ',' StartFcn ',' notebook ')
```

1. 设置通用模块参数

通用模块参数指所有 Simulink 模块都具有的参数。其中模块的回调参数和模型回调参数类似，只不过它是用于每个模块而不是模型。例如，下面的命令为模型中的模块 Sine Wave 设置在窗口中的位置：

```
set _ param([' MyModel ','/',' Sine Wave '],' position ',[20,20,60,80])
```

下面的命令使得当模块 State-Space 被打开（双击鼠标左键）时，为其参数 A 赋 3×3 的随机矩阵：

```
set _ param([' MyModel ','/',' State-Space '],' OpenFcn ',A = randn(3,3)')
```

2. 模块的特定参数

模块的特定参数指每个不同的模块特有的参数。Simulink 通用库下几个子库中的一些常用模块具有特有的参数和取值，但封装模块不能用函数 set _ param 设置参数。

9.6 S－函数的设计和应用

9.6.1 S－函数概述

1. 什么是 S－函数

S－函数，也称之为系统函数(System Functions)，是对动态系统的程序描述。掌握了 S－函数的设计，就可以充分发挥 Simulink 的功能，否则，就只能用模块库里的功能模块去拼出模型，但这往往是不能满足需要的。S－函数可以用 MATLAB 语言或 C 语言编写。用 C 语言编写的 S－函数被 MEX 工具软件编译成 MEX 文件。像其他的 MEX 文件一样，必要时，它可以与 MATLAB 动态地链接。用 MATLAB 语言编写的 S－函数就是一个 M 文件。

S－函数具有很通用的形式，它适用于连续的、离散的和混合的系统。因此，几乎所有的 Simulink 模型都可以用 S－函数描述。

2. S－函数的用途

S－函数的主要用途在于，用户可以利用它创建一个新的通用模型，此模块可以在一个模型中多次使用，而每次使用时，有关参数可以取不同的值。另外，由于 S－函数可以用 C 语言编写，所以可以把已有的 C 程序加到一个仿真程序中。

9.6.2 用 M 文件编写 S－函数

用来定义一个 S－函数的 M 文件，必须能提供系统模型的有关信息。在仿真期间，Simulink 需要这些信息。Simulink、ODE solver(常微分方程或者差分方程数值算法程序)和 S－函数(M 文件)三者之间交互作用，可以完成一些特定的任务。其中包括：确定初始条件和模块的特性参数，

计算导数值(连续状态变量的)、离散状态变量值以及输出变量值。

S－函数的设计步骤很简单,但是可以实现非常复杂的功能:

(1) 写程序。程序的编写有一定的格式,MATLAB 5.3 为用户提供了一个模板,只要在必要的子程序里编写代码并输入参数就可以了。

(2) 从 Function & Tables 子库里把 S－Function 系统的功能模块复制过来,输入程序的文件名,以供调用。

Simulink 提供了一个用 M 文件编写 S－函数的模板。该模板程序存放在 toolbox \ simulink \ blocks 目录下,文件名为 Sfuntmpl.m。下面是把注释部分都去掉后的模板程序。

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 2,
    sys = mdlUpdate(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case 4,
    sys = mdlGetTimeOfNextVarHit(t,x,u);
case 9,
    sys = mdlTerminate(t,x,u);
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 0;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [ ];
str = [ ];
ts = [0, 0];
function sys = mdlDerivatives(t,x,u,)
sys = [ ];
function sys = mdlUpdate(t,x,u)
sys = [ ];
```

```
function sys = mdlOutputs(t,x,u)
sys = [ ] ;
function sys = mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
function sys = mdlTerminate(t,x,u)
sys = [ ];
```

模板程序共有 6 个子程序：

供 Simulink 在仿真的不同阶段调用。这些子程序的前缀为 mdl。用 M 文件表示的 S - 函数，其中设有一个标志参数 flag，每一个 flag 值对应于 S - 函数中的一个子程序。每一次调用 S - 函数时，都要给出一个 flag 值，实际上，它就是执行 S - 函数中与该 flag 值对应的那个子程序。Simulink 在仿真的不同阶段，需要调用 S - 函数中不同的子程序。表 9.8 表明了这种对应关系，其中有些子程序要多次地被调用。编写 S - 函数时，应该根据系统的实际需要，编写相应的子程序调用语句及相应的子程序，并提供必要的参数。

表 9.8 在 S - 函数中调用的子程序及其与 flag 值的对应关系

仿真阶段	S - 函数子程序	flag 值
初始化	mdlInitializeSizes	0
计算连续状态变量导数值	mdlDerivatives	1
更新离散状态变量值	mdlUpdate	2
计算输出值	mdlOutputs	3
计算下一个采样时刻	mdlGetTimeOfNextVarHit	4
结束仿真任务	mdlTerminate	9

mdlInitializeSizes 子程序的功能是初始化。在 S - 函数运行之前，为了使 Simulink 能够识别一个用 M 文件编写的 S - 函数，用户必须向它提供该 S - 函数的有关信息：输入量、输出量、状态变量的个数以及其他特征。

为了向 Simulink 提供这些信息，在子程序 mdlInitializeSizes 的开始处，应调用 simsizes 函数：sizes = simsizes，这个函数返回一个不透明的 size 结构，其结构如下：

① sizes.NumContStates 连续状态变量的个数。

② sizes.NumDiscStates 离散状态变量的个数。

③ sizes.NumOutputs 输出的个数。

④ sizes.NumInputs 输入的个数。

⑤ sizes.DirFeedthrough 直通标志，是否设定为 Direct feedthrough，取决于输出是否为输入的函数，或者取样时间是否为输入的函数。

⑥ sizes.NumSampleTimes 取样时间的个数，一般取 1。

上述结构用 sizes 结构数组来存储，然后用 simsizes 指令去读取。

S - 函数的基本输入参数有 4 个：t，x，u 和 *flag*。其中 t 为仿真时间；x 为状态向量（即使系

统中没有状态变量，也要写 x）；u 为输入向量；*flag* 为标志参数，控制在仿真的各阶段调用S－函数的哪一个子程序。

S－函数的基本输出参数也有4个，它们依次是 *sys*、*x*0、*str* 和 *ts*。其中 *sys* 为一个返回参数的通用符号，它得到何种参数取决于 *flag* 值，比如 *flag* = 3，则 *sys* 得到的是S－函数的输出向量值；*x*0表示初始状态（如果系统中没有状态变量 *x*0将得到一个空阵）；*str* 仅用于方块图同S－函数API（应用程序编程接口）的一致性校验，对于M文件S－函数，它将被置一个空阵；*ts* 是一个两列矩阵，一列是S－函数中各状态变量的采样周期，另一列是相应的采样时间的偏移量，采样周期按递增顺序排列，对于连续系统，采样周期和偏移量都应置成0。

例9.7　利用M文件来写一个限幅积分器的S－函数，并借助S－函数模块来调用此文件。限幅积分器的数学模型如下

$$\dot{x} = \begin{cases} 0, x \leqslant lb, u < 0 \text{ 或 } x \geqslant ub,\ u > 0 \\ u, \text{其他} \end{cases}$$

其中，x 为状态值，u 为输入值，lb 和 ub 分别为积分的下限和上限。

操作步骤如下：

（1）根据数学模型，编写S－函数sfun97.m。

```
function [sys, x0, str, ts] = sfun97(t, x, u, flag, lb, ub, xi)
switch flag,
case 0,
    [sys, x0, str, ts] = mdlInitializeSizes(xi);
case 1,
    sys = mdlDerivatives(t, x, u, lb, ub);
case 2 ,
    sys = mdlUpdate(t, x, u);
case 3,
    sys = mdlOutputs(t, x, u);
case 4,
    sys = mdlGetTimeOfNextVarHit(t, x, u);
case 9,
    sys = mdlTerminate (t, x, u);
otherwise
  error ( [' Unhandled flag = ', num2str (flag) ] );
  end
  function [sys,x0,str,ts] = mdlInitializeSizes(xi)
  sizes  = simsizes;
  sizes.NumContStates = 1;
  sizes.NumDiscStates = 0;
  sizes.NumOutputs = 1;
  sizes.NumInputs = 1;
  sizes.DirFeedthrough = 0;
  sizes.NumSampleTimes = 1;
```

```
sys = simsizes(sizes);
x0 = xi;
str = [ ];
ts = [0, 0];
function sys = mdlDerivatives(t,x,u,lb,ub)
if(x <= lb&u < 0)|(x >= ub&u > 0)
   sys = 0;
else
   sys = u;
end
function sys = mdlUpdate(t,x,u)
sys = [ ];
function sys = mdlOutputs(t,x,u)
sys = x;
function sys = mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1; % Example,set the next hit to be one second later
sys = t + sampleTime;
function sys = mdlTerminate(t,x,u)
sys = [ ];
```

(2) 完成 S－函数的编写后，接着建立 Simulink 模型，将功能模块 S－function 复制到设计区域，打开其参数界面，输入 S－函数的文件名 sfun97.m 和 S－函数的参数 *lb*、*ub*、*xi*，如图 9.34 所示。最后完成的模型如图 9.35 所示。

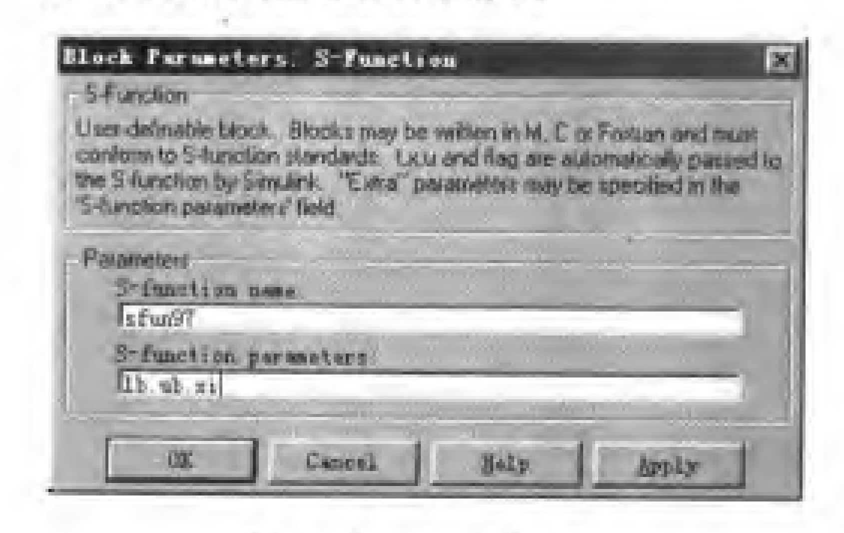


图 9.34 S-Function 参数对话框

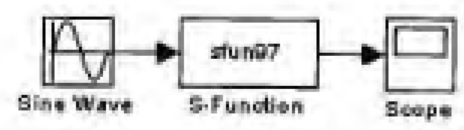


图 9.35 调用 S-Function 的模型

(3) 编写主程序。

```
clear;
lb = -0.5;
ub = 0.5;
xi = 0;
```

```
sim(' sfun97 ');
```

(4) 在命令窗口运行主程序，并双击示波器，即可查看到输入波形及限幅积分的结果。

上面的简单例子中没有状态变量。大多数 S-函数要处理状态变量，可能是连续的或者是离散的，也可能两者都有。

例 9.8　试分别写出一个有连续状态变量的 S-函数，一个有离散状态变量的 S-函数和一个混合系统的 S-函数。

(1) 一个有连续状态变量的 S-函数 csfunc.m。

```
function [sys,x0,str,ts] = csfunc(t,x,u,flag)
%  本 S-函数定义一个用下列连续状态空间方程描述的系统
%  x' = Ax + Bu
%  y = Cx + Du
%  定义矩阵
A = [ -0.09, -0.01; 1, 0];
B = [1, -7;0, -2];
C = [0,2];
D = [ -3,0;1,0];
% 分配标志 flag
switch flag
case 0
        [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D);
case 1
        sys = mdlDerivatives(t,x,u,A,B,C,D);
case 3
        sys = mdlOutputs(t,x,u,A,B,C,D);
case {2,4,9}
        sys = [ ];
otherwise
        error([' Unhandled flag = ',num2str(flag)]);
end
%  csfunc.m 主程序结束
%  以下是子程序
function [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D)
sizes = simsizes;                    %创建 sizes 结构
sizes.NumContStates = 2;             % 以下填充 sizes 结构
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1             %D 阵不是空阵，y 的表达式中含有 u
sizes.NumSampleTimes = 1
sys = simsizes(sizes);               %用 sizes 结构的信息填充数组 sys
x0 = zeros(2,1);                     %零初始条件
```

```
str = [ ];
ts = [0, 0];                          %此例中采样时间是连续性的,故采样周期和偏移时间都设置成 0
%  mdlInitializeSizes 子程序结束
function sys = mdlDerivatives(t,x,u,A,B,C,D)
sys = A * x + B * u
% 返回连续状态变量的导数值
% mdlDerivatives 子程序结束
function sys = mdlOutputs(t,x,u,A,B,C,D)
sys = C * x + D * u;
%返回输出量
% mdlOutputs 子程序结束
```

由于这个 S－函数的 M 文件 csfunc.m 已包含在 Simulink 中,因此,普通的一阶微分方程组都可以使用这个文件来建模。csfunc.m 与内部模块 state-space 类似。这个 S－函数也可作为建立一个具有时变系数的 state-space 模块的出发点。

(2) 一个具有离散状态变量的 S－函数 dsfunc.m。

```
function [sys,x0,str,ts] = dsfunc(t,x,u,flag)
%本 S－函数定义一个用下列离散方程描述的系统
%x (n+1) = Ax(n) + Bu(n)
%y(n) = Cx(n) + Du(n)
%定义矩阵
A = [-1.3839, -0.509; 1.00, 0];
B = [-2.5559, 0; 0, 4.2382];
C = [0, 2.076; 0, 7.7801];
D = [-0.8141, -2.9334; 1.2426, 0]
%分配标志 flag
switch flag
case 0
   [sys, x0,str, ts] = mdlInitializeSizes(A, B, C, D);
case 2
     sys = mdlUpdate(t,x,u,A,B,C,D);
case 3
     sys = mdlOutputs(t,x,u,A,B,C,D);
case {1,4,9}
     sys = [ ];
otherwise
    error(['Unhandledflag = ',num2str(flag)]);
end
%  dsfunc.m 主程序结束
%  以下是子程序
function [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D)
sizes = simsizes;                     %创建 sizes 结构
```

```
sizes.NumContStates = 0;            % 以下填充 sizes 结构
sizes.NumDiscStates = 2;
sizes.NumOutputs = 2;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);              %用 sizes 结构的信息填充数组 sys
x0 = ones(2,1);                     %状态初值
str = [ ];
ts = [1, 0];                        %采用周期 1s,偏移量 0
%  mdlInitializeSizes 子程序结束
function sys = mdlUpdates(t, x, u, A, B, C, D)
sys = A * x + B * u
% mdlUpdate 子程序结束
function sys = mdlOutputs(t, x, u, A, B, C, D)
sys = C * x + D * u;
% mdlOutputs 子程序结束
```

由于本 S－函数的 M 文件 dsfunc.m 已在 Simulink 中,用户可以直接用它为一般的差分方程组建模。这个 S－函数与内部模块 Discrete state-space 类似。用户也可以用它作为建立具有时变系数的离散状态空间系统模型的出发点。

(3) 一个混合系统的 S－函数 mixedm.m。

混合系统是指同时含有连续状态和离散状态的系统。处理混合系统是相当简单的,根据参数 flag 的不同取值,正确地调用 S－函数的连续部分或离散部分的子程序。

编写混合系统的 S－函数(或者任何一个有多种采样周期的 S－函数)应注意的一个问题是 Simulink 在所有的采样时间点上,都要调用子程序 mdlUpdate、mdlOutput 以及 mdlGetTimeOfNextVarHit,这就要求在这些子程序中都应检测当前时刻(即仿真时间)是否是相应变量的采样时刻,若是,则给出该变量的返回值,否则返回空阵,以表示该变量的值尚不变。之所以这样做,是因为在混合系统中,连续状态变量的计算步长总是小于离散部分的采样周期,仿真时间是按照连续状态变量的计算步长累计的。

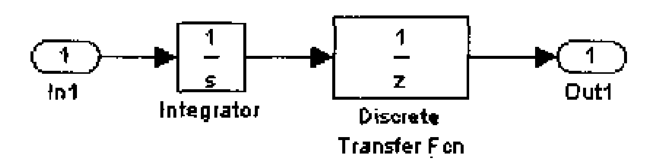


图 9.36　混合系统模型

图 9.36 是由一个积分环节和一个单位延迟环节组成的混合系统,它的 S－函数 M 文件如下:

```
function [sys, x0,str, ts] = mixdem (t, x, u, flag)
%一个积分环节和一个单位延迟环节串联构成的混合系统
dperiod = 1;                        %设置延迟环节的采样周期
doffset = 0;                        %设置偏移量
```

```
switch  flag
case 0
   [sys, x0,str, ts] = mdlInitializeSizes(dperiod, doffset);
case 1
    sys = mdlDerivatives(t, x, u);
case 2
    sys = mdlUpdate(t,x,u, dperiod, doffset);
case 3
    sys = mdlOutputs(t,x,u, dperiod, doffset);
case {4,9}
    sys = [ ];
otherwise
    error(['Unhandled flag = ', num2str(flag)]);
end
%  mixedm.m 主程序结束
%  以下是子程序
function [sys,x0,str,ts] = mdlInitializeSizes(dperiod, doffset)
sizes = simsizes;
sizes.NumContStates = 1;
sizes.NumDiscStates = 1;
sizes.NumOutputs = 1;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 2;          %采用周期两种,连续状态和离散状态各一种
sys = simsizes(sizes);
x0 = ones(2,1);
str = [ ];
ts = [0, 0; dperiod, doffset];     %连续状态的采样周期填0,偏移量也填0。
%  初始化子程序结束
function sys = mdlDerivatives(t, x, u, dperiod, doffset)
sys = u;
%  求导数子程序结束
function sys = mdlUpdate(t, x, u, dperiod, doffset)
if abs(round ( (t-doffset)/dperiod)-(t-doffset)/dperiod) < 1e-8
    sys = x (1);
else
    sys = [ ];
end
```

以上是离散状态更新子程序。其中 t 是仿真时钟(按连续环节的计算步长累计仿真时间)的当前值。如果 t 值减去偏移量之后与 *dperiod* 的整数倍的偏差小于 $dperiod * 1e-8$,则离散状态更新,返回值为 $x(1)$,否则返回空阵,表示离散状态未变。

```
function sys = mdlOutputs(t, x, u, dperiod, doffset)
if abs(round ( (t-doffset)/dperiod)-(t-doffset)/period) < 1e-8
    sys = x (2);
else
    sys = [ ];
end
```

以上是 S－函数输出向量值计算子程序。因为本例中，S－函数的输出即为离散环节的输出，故离散状态更新，则输出量也更新，返回值为 $x(2)$，否则返回空阵，表示输出值未变。

9.6.3　S－函数的命令调用

在仿真过程中，S－函数要被 Simulink 多次调用。其实用 M 文件表示的 S－函数就是一种具有特殊调用格式的函数文件，因此用户也可以直接用 MATLAB 命令调用它。例如，用命令[sizes, x0, xstr, ts] = sfilename([ ], [ ], [ ], 0)，调用名为 *sfilename* 的 S－函数，便可执行它的初始化子程序(这里 flag = 0)，从而获得该 S－函数所描述系统的初始化信息——规模参数 *sizes*、初始条件 *x0*、状态变量所在模块的路径 *xstr*、采样周期及采样时间偏移量 *ts*。

如果把该 S－函数所描述的系统另外用内部模块组成的方块图模型表示，并以名字 *filename* 存盘。再输入 MATLAB 命令[sizes, x0 , xstr, ts] = filename([ ] , [ ], [ ], 0)或[sizes, x0 , xstr, ts] = filename，则得到的结果与上一条命令相同，只是 *xstr* 不再是空阵了，它是一个由字符串元素组成的列向量，各字符串依次表示模型中各状态变量所在模块的路径。这是因为 *filename* 是一种 mdl 文件，它包含了上述 S－函数 M 文件的全部信息，当然也包括初始化信息。

9.7　仿真系统的线性化分析

在一般的非线性系统分析中，常常要在平衡点处提取线性模型。为了解决这一问题，Simulink 提供了 3 个有用的函数：linmod、dlinmod 和 trim。本节将介绍如何使用这 3 个函数。

9.7.1　连续系统的线性化

线性化分析函数 linmod 用来提取非线性系统的近似线性模型。该线性模型可以表示为

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

其中 $\boldsymbol{x}$ 是系统的状态向量，$\boldsymbol{u}$ 是系统的输入向量，$\boldsymbol{y}$ 是系统的输出向量。$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$、$\boldsymbol{D}$ 是常数矩阵，统称为状态空间矩阵。

linmod 函数的调用格式为：

```
[A,B,C,D] = linmod ('sys')
[A,B,C,D] = linmod ('sys', x ,u)
[A,B,C,D] = linmod ('sys', x, u, pert)
[A,B,C,D] = linmod ('sys', x, u, pert, xpert, upert)
```

其中 *sys* 表示需要线性化的模型名称。$\boldsymbol{x}$ 和 $\boldsymbol{u}$ 分别代表状态向量和输入向量。如果定义这些

量，那么就设置了工作点，并在这个点上提取线性模型。*pert* 为可选标量，用来设置 x 和 u 的干扰。如果这个值没有定义，就采样默认值 1e－5。*xpert* 和 *upert* 为可选向量，用来为每一个状态和输入设置扰动。如果定义了这个向量，那么就忽略上面的标量 *pert*，在这种情况下，状态 x 的第 i 个元素受到扰动后就变成 $x(i)$ + *xpert* (i)；输入 u 的第 j 个元素受到扰动后变成 $u(j)$ + *upert* (j)。

说明：

(1) linmod 得到的是用常微分方程描述的 Simulink 模型的线性模型，返回的模型是以状态空间 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ 形式来表示其输入和输出之间的关系，得到在状态向量 $\boldsymbol{x}=\mathbf{0}$ 和输入 $\boldsymbol{u}=\mathbf{0}$ 这个工作点附近的线性化模型。

(2) linmod 是在工作点附近对状态施加扰动后来确定状态导数和输出的变化速率（即雅可比矩阵），并把得到的结果用来计算状态空间矩阵，每一个状态 $\boldsymbol{x}(i)$ 受到扰动后变成

$$\boldsymbol{x}(i)+\Delta(i)$$

其中 $\Delta(i)=\delta(1+|\boldsymbol{x}(i)|)$。

同样，第 j 项输入受到扰动后，变成

$$\boldsymbol{u}(j)+\Delta(j)$$

其中 $\Delta(j)=\delta(1+|\boldsymbol{u}(j)|)$。

linmod 函数不仅可以获取非线性系统的近似线性模型，也可以用来获取线性系统的状态空间模型。例如，从图 9.37 所示的线性系统提取状态空间模型。设系统模型名为 exe9_2，命令如下：

```
[A,B,C,D] = linmod ('exe9_2')
```

命令执行后得到系统的状态空间矩阵

$$\boldsymbol{A}=\begin{bmatrix}-2 & -1 & -1\\ 1 & 0 & 0\\ 0 & 1 & -1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1\\0\\0\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & -1\end{bmatrix},\quad \boldsymbol{D}=\begin{bmatrix}0\\1\end{bmatrix}$$

图 9.37　线性系统模型

9.7.2　离散系统的线性化

dlinmod 能够以任意给定的采样时间对离散系统、多速率系统以及连续和离散混合系统进行线性化。除了第 2 个选项需要插入采样时间来对系统线性化外，dlinmod 的调用格式和 linmod 是

相同的：

```
[A,B,C,D] = dlinmod ('sys', Ts, x,u)
```

调用该函数可以产生一个以状态向量 x 和输入向量 u 为工作点，采样时间为 Ts 的离散状态空间模型。要想得到一个离散系统的近似连续模型，只要把 Ts 设为 0 即可。

如果一个模型是由线性模型、多速率模块、离散模块和连续模块组成的，那么在满足下列条件的情况下，dlinmod 产生一个采样时间为 Ts 且具有相同频率和时间响应的线性模型：

(1) Ts 是系统中所有采样时间的整数倍。

(2) Ts 不小于系统中最慢的采样时间。

(3) 系统是稳定的。

不过，在不满足上面这些条件的情况下，也有可能得到有效的线性模型，但这要看一个系统是否稳定。实际上只要计算线性化后所得矩阵 $\boldsymbol{A}$ 的特征值就可以了。因此，如果 $Ts > 0$ 而且特征值在单位圆内，那么系统就是稳定的。同样，如果 $Ts = 0$，而且所有的特征值在左半平面，那么系统也是稳定的。

当系统不稳定且采样时间不是其他采样时间的整数倍时，dlinmod 就可能产生复矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$。然而在这种情况下，仍然可以通过矩阵 $\boldsymbol{A}$ 的特征值来验证系统的稳定性。dlinmod 可以用来把一个系统的采样时间变成其他值，或者把一个线性离散系统变成一个连续系统。或者反过来，把一个连续系统变成一个离散系统。

例如，对图 9.37 所示的线性系统，取 $Ts = 0.1$，求离散的线性化模型。命令如下：

```
[A,B,C,D] = dlinmod ('exe9_2', 0.1)
```

命令执行后得到离散的线性化状态空间矩阵

$$\boldsymbol{A} = \begin{bmatrix} 0.8142 & -0.0950 & -0.0860 \\ 0.0905 & 0.9952 & -0.0045 \\ 0.0045 & 0.0950 & 0.9047 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0.0905 \\ 0.0047 \\ 0.0002 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

9.7.3 连续系统线性化的一种高级形式

函数 linmod2 提供了一种线性化的高级形式。该函数和 linmod 相比，运行需要更长的时间，但产生的结果要比 linmod 精确。

linmod2 的调用格式和 linmod 调用格式相似，但功能不同。linmod2 设法平衡舍入误差（由于小扰动引起的，并与计算精度有关的误差）与截断误差（由于大扰动引起的，使分段线性近似无效而产生的误差），并在两者之间进行折衷。

对于调用命令[A,B, C,D] = linmod2('sys', x, u, pert)，其中 *pert* 表示可以施加的最小扰动，默认值为 1e-8。linmod2 的优点是能够探测到不连续性，并产生一个警告信息。例如

```
warning: discontinuity detected ad A (2, 3)
```

当出现上述这样的警告信息时，可在其他的工作点附近进行试验来得到线性化模型。

例如，对图 9.37 用 linmod2 再解一次，输入命令

```
[A,B,C,D] = linmod2('exe9_2')
```

得到和运用 linmod 函数一样的结果。

关于模型线性化的几点说明：

(1) 若被线性化的系统本身是线性的,那么线性化过程中不存在截断误差,扰动可取任意大小,工作点也不影响所得结果。一般来说,扰动取值较大,有利于减小圆整误差。

(2) 在线性化过程中,模型的状态次序不变。

(3) 在 Simulink 框图上,系统的输入输出要依次编号。且系统的输入和输出必须用 Singnals & Systems 库中的 In1 和 Out1 模块来定义。Singnal Generator 和 Scope 模块不能作为系统的输入和输出。

9.7.4 平衡分析

在给定输入、输出及状态条件下,Simulink 提供的函数 trim 可以用来确定系统的稳态平衡点。其调用格式为:

```
[x,u,y,dx] = trim('sys')
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy)
```

其中 ***x***, ***u***, ***y***, ***dx*** 分别代表状态向量、输入向量、输出向量和状态向量的导数。*sys* 是模型名。***x*0**, ***u*0**, ***y*0** 分别为状态向量 ***x***、输入向量 ***u*** 和输出向量 ***y*** 的初始猜测值,它们的输入格式必须是列向量形式。***ix*** 是向量,它的元素是那些在寻找过程中,要求保持其值固定不变(等于初始猜测值)的那些状态变量的序号。***iu***、***iy*** 与 ***ix*** 类似,分别适用于 ***u*** 和 ***y***。

trim 的功能是试图找到输入 ***u*** 和状态 ***x*** 的值来使状态的导数为 0,这样的工作点被称作平衡点,即系统在稳态时的工作点。既然这样的平衡点通常不是惟一的,因此有必要对状态 ***x***、输入 ***u*** 和输出 ***y*** 的值进行固定。

命令[x,u,y] = trim('sys')的功能是设法找到一个平衡点,使[x;u;y]的绝对值达到最小。命令[x,u,y] = trim('sys',x0,u0,y0)使[x - x0;u - u0;y - y0]的绝对值达到最小。

x, ***u***, ***y*** 的每一个元素可用下面的调用格式进行固定:

```
trim('sys',x0,u0,y0,ix,iu,iy)
```

整数向量 ***ix***, ***iu*** 和 ***iy*** 用来指定 ***x*0**, ***u*0** 和 ***y*0** 中保持不变的分量下标,使搜索受约束进行。既然无法保证平衡点一定存在,因此问题就转换为寻找一个稳态值,使得

```
abs([x(ix) - x0(ix);u(iu) - u0(iu);y(iy) - y0(iy)])
```

绝对值达到最小。

trim 在限制状态导数为零的情况下,用一个有约束的优化算法来求解一个由 ***x***, ***u*** 和 ***y*** 的希望值所组成的绝对值最小问题。对于这样一个问题,有可能没有可行解。在这种情况下,trim 就在状态导数偏离 0 这种最坏条件下,使上面的绝对值达到最小。

要使导数固定为一个非 0 值,可以采用

```
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy,dx0,idx)
```

其中 *dx*0 表示希望的偏离值,*idx* 用来表示对 ***dx*** 中的哪一个元素进行固定。

下面仍以图 9.37 的模型为例。用 trim 函数来找到使输出为 1 时输入和状态所对应的值。

首先输入 ***u*** 和状态变量 ***x*** 作为初始假设,然后设置输出 ***y*** 的希望值:

```
x = [0; 0; 0];
u = 0;
y = [1;1];
```

再用索引变量指出哪个变量是固定的，哪个变量是可以变化的。例如

```
ix = [ ];            % 没有固定任何状态变量
iu = [ ];            % 没有固定输入
iy = [1; 2];         % 固定输出端口 1 和输出端口 2
```

最后调用 trim 函数：

```
[x,u,y,dx] = trim('exe9_2,[ ],[ ],y,[ ],[ ],iy)
```

得到状态向量 $\boldsymbol{x}$、输入向量 $\boldsymbol{u}$、输出向量 $\boldsymbol{y}$ 以及状态向量的导数 $\boldsymbol{dx}$ 分别为：

$$\boldsymbol{x} = \begin{bmatrix} 0.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}, \boldsymbol{u} = 2, \boldsymbol{y} = \begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix}, \boldsymbol{dx} = 1.0\mathrm{e}-015 * \begin{bmatrix} -0.2220 \\ -0.0227 \\ 0.3331 \end{bmatrix}$$

在平衡工作点，$\boldsymbol{dx}$ 本应等于零，这里 $\boldsymbol{dx} \neq \boldsymbol{0}$，是由于计算误差或者搜索精度决定的。

除非问题本身的最小值惟一，否则不可能保证所求得的平衡点是最佳值。因此，要想找到全局的最佳平衡点，多尝试几组初始值是必要的。

习　题　九

1. 建立如图 9.38 所示的方框图模型，用菜单方式运行该模型的仿真。改变 Gain 模型的增益，看 Scope 显示波形的变化。用 Silder Gain 模型取代 Gain 模块，重复上述操作。

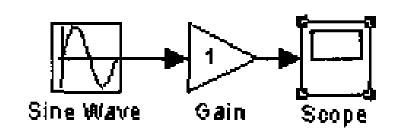


图 9.38　习题九第 1 题图

2. 建立如图 9.39 所示的方框图模型。用菜单方式进行仿真，改变 Slider Gain 模型的增益，观察 $x-y$ 波形的变化，用浮动的 Scope 模块观测各点波形。

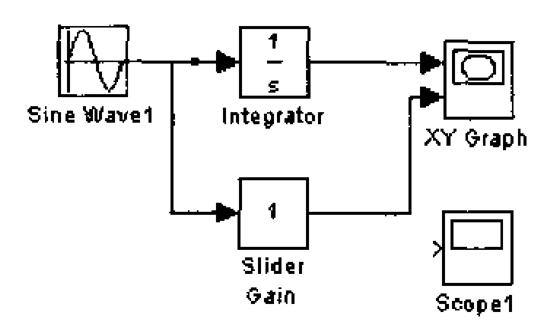


图 9.39　习题九第 2 题图

3. 将图 9.38 中的 Scope 模块换成 Outport 模块。在 Simulink Parameter 对话框中把时间和输出选作返回变量，分别填以变量名 t 和 y。用菜单方式运行仿真，然后用绘图命令画出曲线 $y-t$。

4. 用两个 Outport 模块取代图 9.39 中的 XY Graph 模块。在 Simulation Parameter 对话框中，把时间和输出选作返回变量，分别填以变量名 t 和 $[y_1, y_2]$。用菜单方式进行仿真，然后用绘图命令画出曲线 y_1-t，y_2-t 和 y_1-y_2。

5. 用 To Workspace 模块取代第 3 题中的 Outport 模块。用菜单方式进行仿真，然后画曲线。

6. 对第 3 题的 Simulation Parameter 对话框只解变算法(Solver)，例如采用 ode45、ode23 变步长方法，其他参数保持不变，看仿真结果有何差别。再改用定步长的 Diserete 算法，把仿真结果与前面几种方法结果对比，能得出什么结论？把变步长算法产生的计算时间点作为定步长 Discrete 算法时的输出时间点，重新用 Discrete 算法进行仿真，然后在同一坐标平面上绘制两种算法仿真结果的数据点曲线。

7. 用 MATLAB 命令仿真习题 1、习题 2。

8. 简述 3 种条件执行子系统的特点。

9. 设计一个实现下面函数模块的子系统

$Output = (input1 + input2) \times input3 - input4$

然后对子系统进行封装。要求对 Icon、Initialization 和 Documentation 3 个参数选项卡进行设置。

10. 若典型反馈控制系统的模型为

$$G(S) = \frac{1}{S(S^2/2600 + S/26 + 1)}, \quad G_C(S) = K, \quad H(S) = \frac{1}{0.04S + 1}$$

(1) 试对不同的 K 值绘制出闭环系统单位阶跃响应曲线，并用试探法找出使得闭环系统临界不稳定的增益 K 值，并用根轨迹法检验所得出的结果。

(2) 令 $K = 4$，显示系统的脉冲响应曲线。

第 10 章　MATLAB 应用实例

通过前面的学习，读者应该有这样的体会，MATLAB 的确是一种优秀的科学计算语言，在需要进行大量矩阵计算、各种数值处理、符号计算以及图形处理的各个应用领域，都具有很大的应用价值。用 MATLAB 来解决诸多专业领域的实际问题，能够大大地提高工作效率和质量。

本章介绍一些 MATLAB 应用实例，目的是让读者通过这些实例，进一步掌握 MATLAB 程序设计的基本方法，并且了解 MATLAB 在相关领域的应用价值，从而引导读者更好地应用 MATLAB 来解决自己专业领域的实际问题。本章给出的实例尽量结合一定的专业背景，但覆盖面是有限的。所以，本章内容仅起一个引导、提示的作用，希望能抛砖引玉。另外，在实际应用中，可能还要用到一些学科性工具箱，因其数目繁多，本书不可能一一涉及，读者如有需要，可参考有关专著。

10.1　MATLAB 在电路分析中的应用

10.1.1　概述

1. 矩阵计算与线性电路分析

矩阵工具引入电路理论已有半个多世纪的历史。矩阵的引入使电路定律的表述更为精炼。由于把多变量的系统在形式上按单变量表示，整个理论显得更为简约，概念更为清晰，而且能从整体上掌握电路的状态。传统的克希霍夫定律、支路电流法、回路电流法以及节点电压法都可以以矩阵形式出现。

矩阵是 MATLAB 最基本的数据对象，MATLAB 的大部分运算或命令都是在矩阵运算的意义下执行的，而且 MATLAB 的矩阵运算定义在复数域上，这为电路分析带来了方便。

2. 微分方程求解与电路瞬态分析

当动态电路从某一稳定状态转换到另一稳定状态时，有些物理量(如 u_C，i_L，q，ψ)并不是立即突变的，而是需要一定的时间。在这期间，电路将呈现出和稳定状态不同的特别现象，这种现象为电路的过渡过程或瞬态现象。分析电路的瞬态现象时，可以建立关于电压和电流的微分方程，再按所给定的初始条件来进行求解。

MATLAB 提供了常微分方程初值问题的数值解法，利用有关函数可进行电路瞬态分析。此外，还可以利用 MATLAB 符号计算或 Simulink 仿真来求解。

3. 图形功能与电路分析

利用 MATLAB 的图形功能可以绘制电路的各种响应曲线。MATLAB 的图形功能很强并可对其实行控制。

10.1.2 实例

例 10.1 三相不平衡交流电路分析。计算图 10.1 所示三相不平衡交流电路各支路电流($\dot{I}_a,\dot{I}_b,\dot{I}_c$)并绘制相量图。其中 $r_a=r_b=r_c=5\ \Omega, r_{ab}=6\ \Omega, r_{bc}=10\ \Omega, r_{ca}=15\ \Omega, E=220\ \text{V}$。

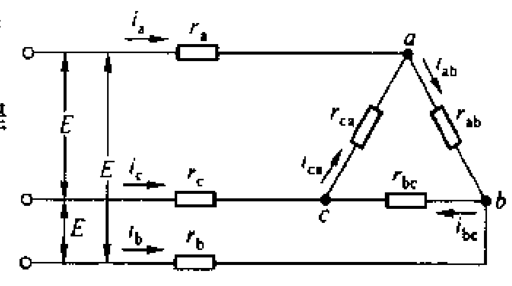


图 10.1 三相不平衡交流电路

根据克希霍夫定律并代入数值,$\dot{I}_{ab},\dot{I}_{bc}$和$\dot{I}_{ca}$满足方程组

$$\begin{cases}16\dot{I}_{ab}-5\dot{I}_{bc}-5\dot{I}_{ca}=220\\-5\dot{I}_{ab}+20\dot{I}_{bc}-5\dot{I}_{ca}=-110-\text{j}110\sqrt{3}\\-5\dot{I}_{ab}-5\dot{I}_{bc}+25\dot{I}_{ca}=-110+\text{j}110\sqrt{3}\end{cases}$$

用 MATLAB 先求 $\dot{I}_{ab},\dot{I}_{bc}$和$\dot{I}_{ca}$,进而求各支路电流:

```
a=[16,-5,-5;-5,20,-5;-5,-5,25];          %系数矩阵
b=[220;-110-110*sqrt(3)*i;-110+110*sqrt(3)*i];
I=inv(a)*b;                              %解方程
Ia=I(1)-I(3)                             %求各支路电流
Ib=I(2)-I(1)
Ic=I(3)-I(2)
h=compass([Ia,Ib,Ic]);                   %绘制相量图
set(h,'LineStyle',3);
```

运行程序,得到各支路电流为:

```
Ia =
  14.5783 - 6.5804i
Ib =
  -15.1084 - 7.4986i
Ic =
  0.5301 + 14.0790i
```

各支路电流相量图如图 10.2 所示。

例 10.2 调谐振荡电路分析。分析图 10.3 所示的调谐振荡电路($i_L=f(v)=\alpha+\beta v-\gamma v^3$, $\beta>0,\gamma>0$),要求绘制振荡波形和相轨迹。

根据克希霍夫定律,图 10.3 调谐振荡电路的电流 i 满足下式(忽略流入放大元件的电流)

$$L\frac{\mathrm{d}i}{\mathrm{d}t}+ri+\frac{1}{C}\int i\mathrm{d}t=M\frac{\mathrm{d}i_L}{\mathrm{d}t}$$

$$i=C\frac{\mathrm{d}v}{\mathrm{d}t}$$

由此两式可得到

$$LC\frac{\mathrm{d}^2v}{\mathrm{d}t^2}+rC\frac{\mathrm{d}v}{\mathrm{d}t}+v=M\frac{\mathrm{d}i_L}{\mathrm{d}t}$$

式中

$$i_L=f(v)=\alpha+\beta v-\gamma v^3\quad,\beta>0,\gamma>0$$

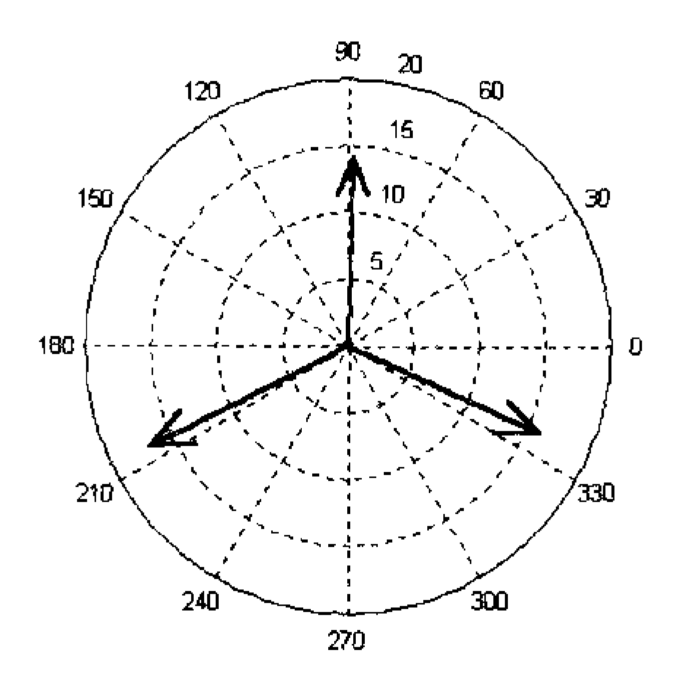


图 10.2　各支路电流相量图

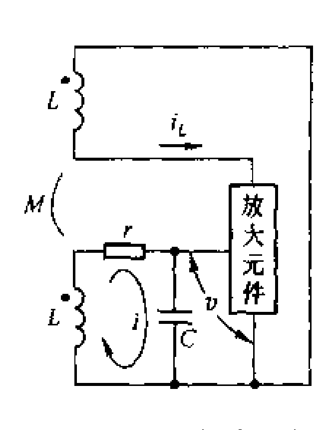


图 10.3　调谐振荡电路

再将上式写成

$$LC\frac{d^2v}{dt^2}+(rC-M\beta+3M\gamma v^2)\frac{dv}{dt}+v=0$$

用 $t=\sqrt{LC}\tau$ 作变换,可得到

$$\frac{d^2v}{d\tau^2}+\frac{1}{\sqrt{LC}}(rC-M\beta+3M\gamma v^2)\frac{dv}{d\tau}+v=0$$

如果 $rC-M\beta<0$,并设

$$\delta=\frac{3M\gamma}{\sqrt{LC}},\mu=\frac{M\beta-rC}{\sqrt{LC}}$$

则有

$$\frac{d^2v}{dt^2}+(\delta v^2-\mu)\frac{dv}{dt}+v=0$$

设 $v=\sqrt{\frac{\mu}{\delta}}y$,上式可写成

$$\frac{d^2y}{dt^2}+\mu(y^2-1)\frac{dy}{dt}+y=0$$

这是著名的范德波尔(Van der Pol)方程。它是一个非线性方程,利用 MATLAB 可以求其数值解。设其初始条件为:$t=0,y=a,\frac{dy}{dt}=b$。MATLAB 求常微分方程初值问题数值解的函数是对一阶常微分方程组设计的,因此对高阶常微分方程,需先将它转化为一阶常微分方程组,即状态方程。选择状态变量 $y_1=\frac{dy}{dt},y_2=y$,则可写出 Van der Pol 方程的状态方程形式:

$$\frac{dy_1}{dt}=\mu(1-y_2^2)y_1-y_2$$

$$\frac{dy_2}{dt}=y_1$$

基于以上状态方程建立函数文件 vdpol.m：

```
function ydot = vdpol(t,y)
ydot(1) = 0.1 * (1 - y(2)^2) * y(1) - y(2);      %μ的值可以任意变化,此处取 0.1
ydot(2) = y(1);
ydot = ydot ';
```

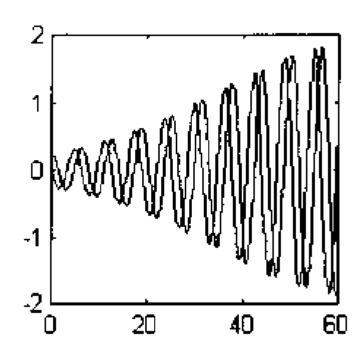
求解微分方程：

```
t0 = 0;tf = 60;                          %确定积分区间
y0 = [0;0.25];                           %确定初始条件
[t,y] = ode45(' vdpol ',[t0,tf],y0);     %求解微分方程
```

绘制振荡波形$(t, dy/dt)$、(t, y)和相轨迹$(y, dy/dt)$：

```
subplot(1,2,1);plot(t,y);
subplot(1,2,2);plot(y(:,2),y(:,1));
```

图 10.4 是 μ 取 0.1 时的波形图和相轨迹，从中可以清楚地了解起振时的波形变化。当 μ 值很小时振荡波形和正弦波接近，相轨迹极限环接近圆形。随着 μ 值的增大，振荡波形将有显著失真，相轨迹变形(图 10.5)。



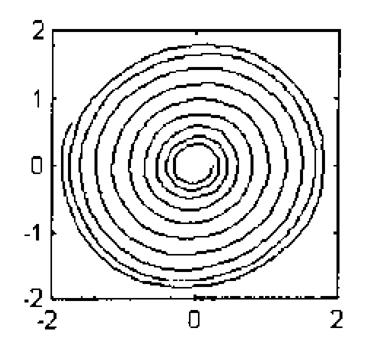
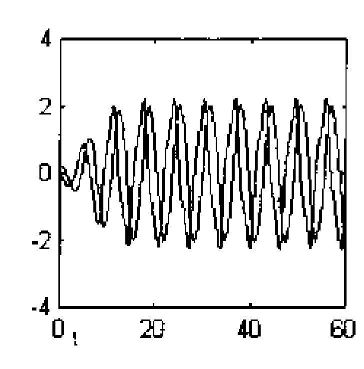


图 10.4 μ 取 0.1 时的振荡波形和相轨迹



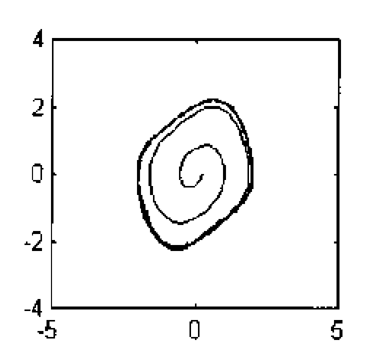


图 10.5 μ 取 0.5 时的振荡波形和相轨迹

MATLAB 定义在复数域上的矩阵计算、微分方程求解以及图形功能为电路分析创造了良好的条件。利用 MATLAB 来进行电路分析比传统方法更为简洁、高效。

10.2 MATLAB 在控制系统分析中的应用

10.2.1 概述

以前要想获得一个控制系统的响应，往往需要编写一个数值计算程序，如果还想绘制系统的响应曲线，还得编写一个画图子程序。如用一些传统高级语言编程，一般情况下求解一个简单的问题（如一个系统的阶跃响应），也需要花费很多的时间，而且得到的结果有时不一定十分满意。MATLAB 语言出现以后，特别是 MATLAB 的控制系统工具箱（Control System Toolbox）及 Simulink 仿真软件问世以来，确实给系统分析者带来了福音，因为用户不必像以前那样将时间和精力无谓地花费在一些无关紧要的工作中去。

用 MATLAB 进行控制系统计算机辅助分析，使得以往十分困难的系统仿真问题可以轻而易举地解决。本节应用 MATLAB 控制系统工具箱和 Simulink 仿真软件来对线性时不变（LTI）系统进行仿真。

系统仿真实质上就是对描述系统的数学模型进行求解。对控制系统来说，系统的数学模型实际上是某种微分方程或差分方程模型，因而在仿真过程中需要以某种数值算法从给定的初始条件出发，逐步地算出每一个时刻系统的响应，最后绘制出系统的响应曲线，由此来分析系统的性能。

MATLAB 的控制系统工具箱提供了很多对线性系统在特定输入下进行仿真的函数。例如连续时间系统在阶跃输入激励下的仿真函数 step、在脉冲激励下的仿真函数 impulse 等，其中阶跃响应函数 step 的调用格式为：

```
step(sys)或[Y,T] = step(sys)
```

其中 *sys* 代表 LTI 的传递函数模型，第一种调用方式直接绘制出阶跃响应曲线，而第二种调用方式不绘制曲线，而返回输出响应 Y 和时间向量 T。

求取脉冲响应的函数 impulse 和 step 函数的调用格式完全一致。

10.2.2 实例

例 10.3 --典型线性反馈控制系统结构如图 10.6 所示，图中 $R(S)$ 为输入函数，$Y(S)$ 为输出，$G_C(s)$ 为控制器模型，$G(s)$ 为对象模型，$H(s)$ 为反馈模型。各个模块分别为

$$G(s)=\frac{4}{S^3+2S^2+3S+4},\quad G_C(s)=\frac{S-3}{S+3},\quad H(s)=\frac{1}{0.01S+1}$$

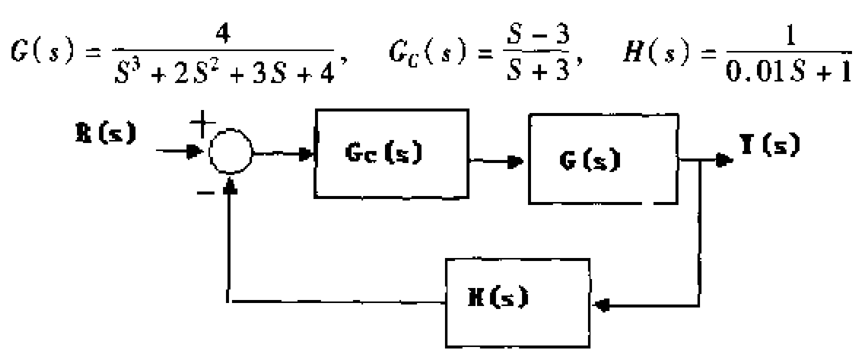


图 10.6 线性反馈控制系统结构

试分别利用 MATLAB 控制系统工具箱提供的仿真函数和 Simulink 仿真软件求出开环和闭环系统的阶跃响应曲线。

解法 1 利用 MATLAB 控制系统工具箱中已经定义的一些 LTI 仿真函数，编写如下程序：

```
G = tf(4,[1,2,3,4]);Gc = tf([1, -3],[1,3]);H = tf(1,[0.01,1]);
G_o = Gc * G;                  %构造开环系统的传递函数。
G_c = feedback(G_o,H);         %构造闭环系统的传递函数。
step(G_o);                     %求开环系统的阶跃响应并绘制相应的曲线。
figure;step(G_c);              %求闭环系统的阶跃响应并绘制相应的曲线。
```

说明：

(1) tf 函数是控制系统工具箱中的一个 LTI 对象函数，由给定的传递函数分子分母构造出单个的传递函数，使系统模型的输入和处理更加方便。该函数的调用格式为：

```
G = tf(num,den)
```

其中 num、den 分别为系统的分子和分母系数向量，返回的变量为系统传递函数。

(2) 控制系统工具箱提供的 feedback 函数，用来求取反馈连接下总的系统模型。该函数的调用格式为：

```
G = feedback(G1,G2)
```

其中 $G1$ 和 $G2$ 分别为前向模型和反向模型的 LTI 对象，而 G 为总系统模型。

程序执行后，得到开环系统和闭环系统的阶跃响应曲线分别如图 10.7 和图 10.8 所示。

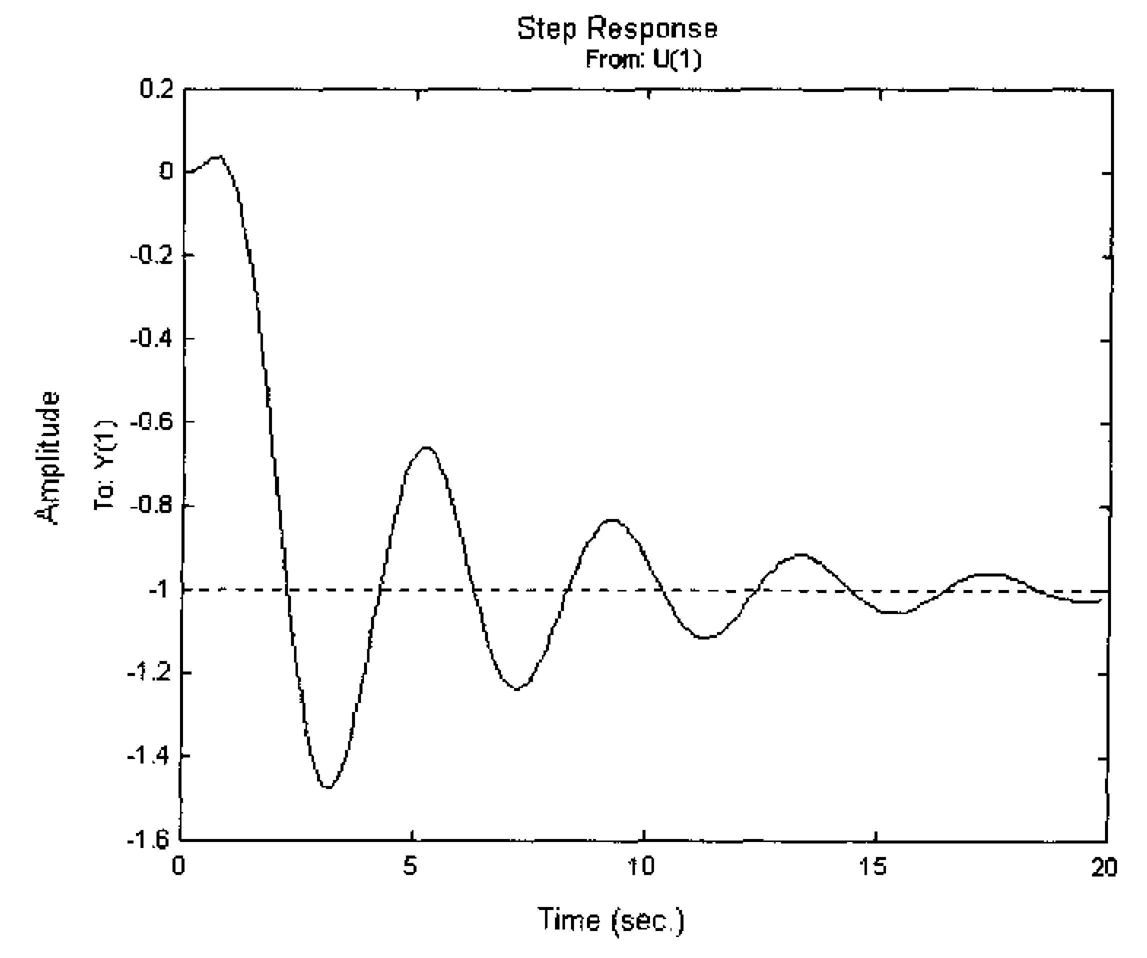


图 10.7 开环系统的阶跃响应曲线

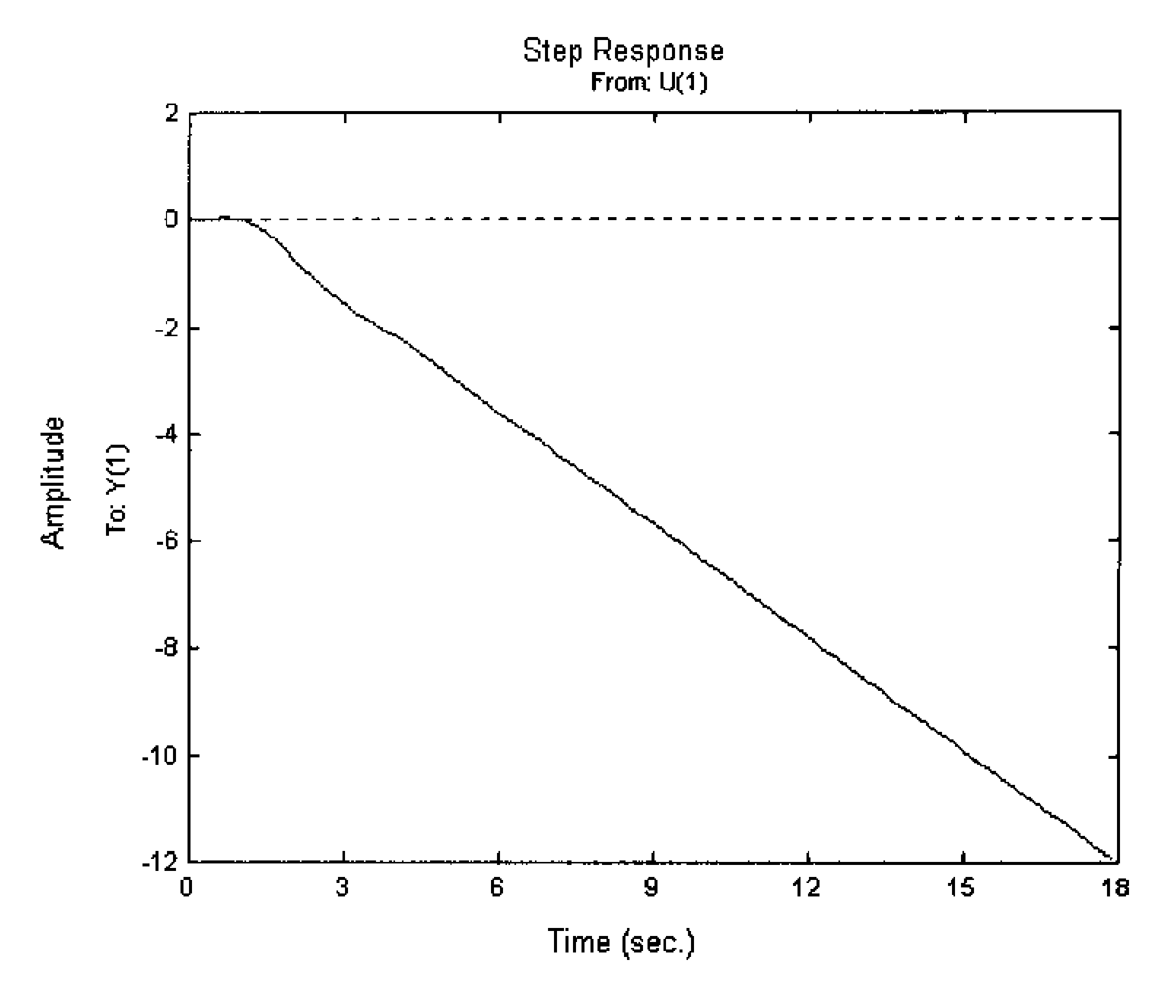


图 10.8 闭环系统的阶跃响应曲线

解法2 利用 Simulink 仿真软件进行分析，步骤如下：

(1) 打开 MATLAB 模块库浏览器窗口以新建一个模型窗口。

(2) 在模块库浏览器窗口中双击 Control System Toolbox 图标，即打开控制系统工具箱，并将其中的 LTI 模型复制 3 个至新建的模型窗口，分别作为 $G_c(s)$、$G(s)$和 $H(s)$的函数模块。注意：由于 $H(s)$是反向模块(即表示负反馈)，所以需按快捷键 Ctrl + R 改变其传输方向。

(3) 双击其中的 $G(s)$模块，将出现设置模块参数对话框，将 LTI system variable 一栏中原来系统默认的传递函数 tf(1,[1,1])修改为 tf(4,[1,2,3,4])，然后对 $G_c(s)$、$H(s)$也分别做相应修改。

(4) 将模块库浏览器窗口中的 Step 模块、Sum 模块以及 Scope 模块分别复制至模型窗口，并按图 10.9 连接好系统。

(5) 选择模型窗口 Simulation 菜单中的 Start 命令，即可得到与图 10.8 完全一致的闭环系统阶跃响应曲线。

(6) 断开图 10.9 中 $H(s)$模块左侧或右侧的连线，使其成为开环系统，再进行仿真，即可得到与图 10.7 完全一致的开环系统阶跃响应曲线。

从这个例子可以看出，开环系统是稳定的，而闭环系统是不稳定的。因此，并不是所有的控制器和闭环结构都能够改善原系统的性能，事实上，如果控制器设计不当，则将使闭环系统的特性恶化。

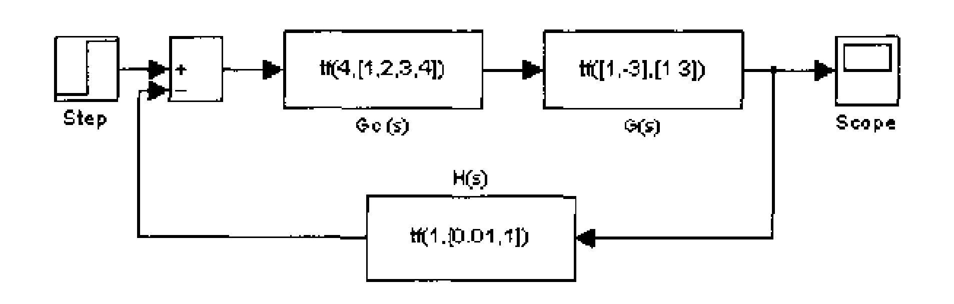


图 10.9 线性反馈控制系统仿真模型

无论是运用 MATLAB 控制系统工具箱的 LTI 仿真函数,还是利用 Simulink 仿真软件,进行线性系统时域响应分析都是十分高效、便捷的。当然 MATLAB 在控制系统中的应用远非如此,其功能是十分强大的,有兴趣的读者可以参考相关文献。

10.3 MATLAB 在数学建模中的应用

10.3.1 概述

计算机的广泛应用和迅速普及,促成了数学建模的发展,也促成了实验数学的诞生。近 20 年来,国内外数学工作者一直在讨论实验数学的发展问题,并认识到实验数学应当与纯粹数学、应用数学鼎足而立。

MATLAB 强大的计算与图形功能为以实验的方式学习和研究数学理论创造了良好的条件,成为数学工作者一个强有力的工具。数学中的许多抽象定理和结论,如今可以在实验中一目了然,新思想、新方法也可以在计算机上迅速地呈现。

一年一度的全国大学生数学建模竞赛,不论是对提高学生的数学素质和计算机应用能力,还是培养其从事科学研究的能力,都起着非常重要的作用。数学建模竞赛是 20 世纪 80 年代中期美国率先进行并受到国际上众多国家响应的一种大学生数学竞赛。我国数学建模竞赛始于 1993 年,竞赛试题有很强的实际应用背景,没有惟一答案,要求参赛的 3 个队员充分发挥集体智慧,在 72 小时内对试题给出一个尽可能完整合理的解答,包括查阅资料,了解有关领域知识,建立数学模型,研究算法,进行计算机编程和运算,得出结论,进行必要的分析,最后以书面报告形式把所有结果表述出来。这种竞赛实质上类似于一项科研课题的研究,对参赛队员的总体素质,包括专业知识、数学建模能力、计算机应用能力、文字表达能力以及集体协作精神都是严峻的考验。下面就数学建模与数学实验,介绍 MATLAB 的应用。

10.3.2 实例

例 10.4 自行车轮饰物的运动轨迹。为了使平淡的自行车增添一份美感,同时,也为了增加自行车的安全系数,一些骑车的人及自行车厂家在自行车的辐条上安装一款亮丽夺目的饰物,当有这种饰物的自行车在马路上驶过时,这种饰物就像游龙一样,对街边的行人闪过一道波浪形

的轨迹。这一波一闪的游龙,其轨迹是什么曲线?试画出它的图形。当自行车在一个抛物线型的拱桥上通过时,或是在一拱一拱的正弦曲线上通过时,这个轨迹是什么曲线?试画出其图形。

假设自行车在运行过程中,始终只与曲线 $y=f(x)$的一个点接触。将路面视为一通过原点的曲线 $f(x)$,车轮视为一半径为 R 的圆,该圆位于曲线的上方且与此曲线相切。设饰物 P 在离轮心距离为 r 处,显然 $r\leqslant R$。OP 与圆交于 A,现以 A 为初始接触点,圆(即车轮)滚过 θ 角以后(假定车轮和路面作无滑动的滚动),圆与曲线的接触点变为 B,圆心从 O 移到 O',饰物 P 从原位置移到 P',如图 10.10 所示。

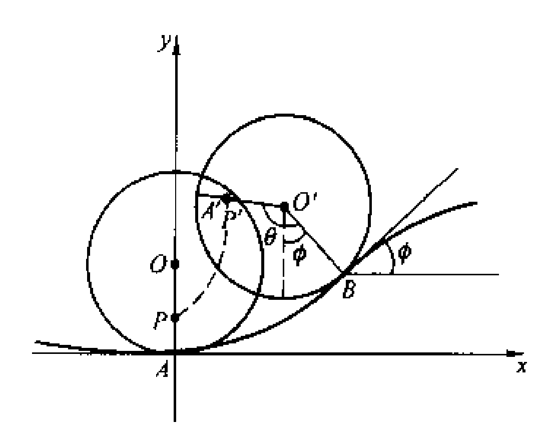


图 10.10　车轮运动示意图

设 $B(x_0,y_0)$, $O'(X,Y)$, $P'(x,y)$,则

$$BA' = R\theta = BA = \int_0^{x_0}\sqrt{1+f'(x)^2}\,\mathrm{d}x$$

所以

$$\theta = \frac{1}{R}\int_0^{x_0}\sqrt{1+f'(x)^2}\,\mathrm{d}x$$

$O'(X,Y)$在 $y=f(x)$过 B 点的法线的上侧,且在以 B 为圆心的圆周上,所以有

$$Y-f(x_0) = -\frac{1}{f'(x_0)}(X-x_0)$$

$$(X-x_0)^2+(Y-f(x_0))^2 = R^2$$

联立解得

$$X = x_0 - \frac{Rf'(x_0)}{\sqrt{1+f'(x_0)^2}}$$

$$Y = f(x_0) + \frac{R}{\sqrt{1+f'(x_0)^2}}$$

又对 $P'(x,y)$,有

$$x = X - r\sin(\theta-\phi),\ y = Y - r\cos(\theta-\phi)$$

其中 $\phi=\mathrm{atan}(f'(x_0))$, $\theta = \frac{1}{R}\int_0^{x_0}\sqrt{1+f'(x)}\,\mathrm{d}x$。所以,饰物的运动轨迹方程为

$$x = x_0 - \frac{Rf'(x_0)}{\sqrt{1+f'(x_0)^2}} - r\sin(\theta-\phi)$$

$$y = f(x_0) + \frac{R}{\sqrt{1+f'(x_0)^2}} - r\cos(\theta-\phi)$$

(1) 取 $f(x)=0$,则有 $\theta=x_0/R$, $\phi=0$,这时饰物的运动轨迹方程为

$$x = x_0 - r\sin(x_0/R),\ y = R - r\cos(x_0/R)$$

(2) 取 $f(x)=0.2-0.2x^2$,则有 $\theta = \frac{1}{R}\int_0^{x_0}\sqrt{1+(-0.4x)^2}\,\mathrm{d}x$, $\phi=\mathrm{atan}(-0.4x_0)$,这时饰物的运动轨迹方程为

$$x = x_0 + \frac{0.4Rx_0}{\sqrt{1+(-0.4x_0)^2}} - r\sin(\theta-\phi)$$

$$y = 0.2 - 0.2x_0^2 + \frac{R}{\sqrt{1+(-0.4x_0)^2}} - r\cos(\theta - \phi)$$

(3) 取 $f(x) = 0.3\sin x$，则有 $\theta = \frac{1}{R}\int_0^{x_0}\sqrt{1+(0.3\cos x)^2}\mathrm{d}x$，$\phi = \mathrm{atan}(0.3\cos x_0)$，这时饰物的运动轨迹方程为

$$x = x_0 - \frac{0.3R\cos x_0}{\sqrt{1+(0.3\cos x_0)^2}} - r\sin(\theta - \phi)$$

$$y = 0.3\sin x_0 + \frac{R}{\sqrt{1+(0.3\cos x_0)^2}} - r\cos(\theta - \phi)$$

程序如下：

```
clear;
%第(1)类情况的实现
x0 = 0:0.01:2;R = 0.1;r = 0.075;
x1 = x0 - r * sin(x0/R);                                  %计算 f(x) = 0 时 p 点运动轨迹
y1 = R - r * cos(x0/R);
subplot(3,1,1);
plot(x1,y1,x0,0);                                         %绘制运动轨迹曲线和 f(x)曲线
grid on
%第(2)类情况的实现
x0 = -1:0.01:1;R = 0.1;r = 0.1;
y0 = 0.2 - 0.2 * x0.^2;                                   %计算路面曲线
fai = atan( - 0.4 * x0);                                  %求 φ
int = inline(' sqrt(1 + ( - 0.4 * x).^2)');               %定义 θ 的积分函数
for k = 1:length(x0)
        theta1(k) = quad(int,0,x0(k))/R;                  %调用 quad 函数求 θ
end
x2 = x0 + R * 0.4 * x0./sqrt(1 + ( - 0.4 * x0).^2) - r * sin(theta1 - fai);   %p 点运动轨迹方程
y2 = y0 + R./sqrt(1 + ( - 0.4 * x0).^2) - r * cos(theta1 - fai);
subplot(3,1,2);
plot(x2,y2,x0,y0)                                         %绘制运动轨迹曲线和 f(x)曲线
grid on
%第(3)类情况的实现
x0 = 0:0.01:10;R = 0.1;r = 0.075;
y0 = 0.3 * sin(x0);                                       %计算路面曲线
fai = atan(0.3 * cos(x0));                                %求 φ
int = inline(' sqrt(1 + (0.3 * cos(x)).^2)');             %定义 θ 的积分函数
for k = 1:length(x0)
        theta2(k) = quad(int,0,x0(k))/R;                  %调用 quad 函数求 θ
end
x3 = x0 - 0.3 * R * cos(x0)./sqrt(1 + (0.3 * cos(x0)).^2) - r * sin(theta2 - fai);   %p 点运动轨迹方程
y3 = 0.3 * sin(x0) + R./sqrt(1 + (0.3 * cos(x0)).^2) - r * cos(theta2 - fai);
```

```
subplot(3,1,3);
plot(x3,y3,x0,y0)
grid on
```

运行程序得图 10.11 所示曲线图。

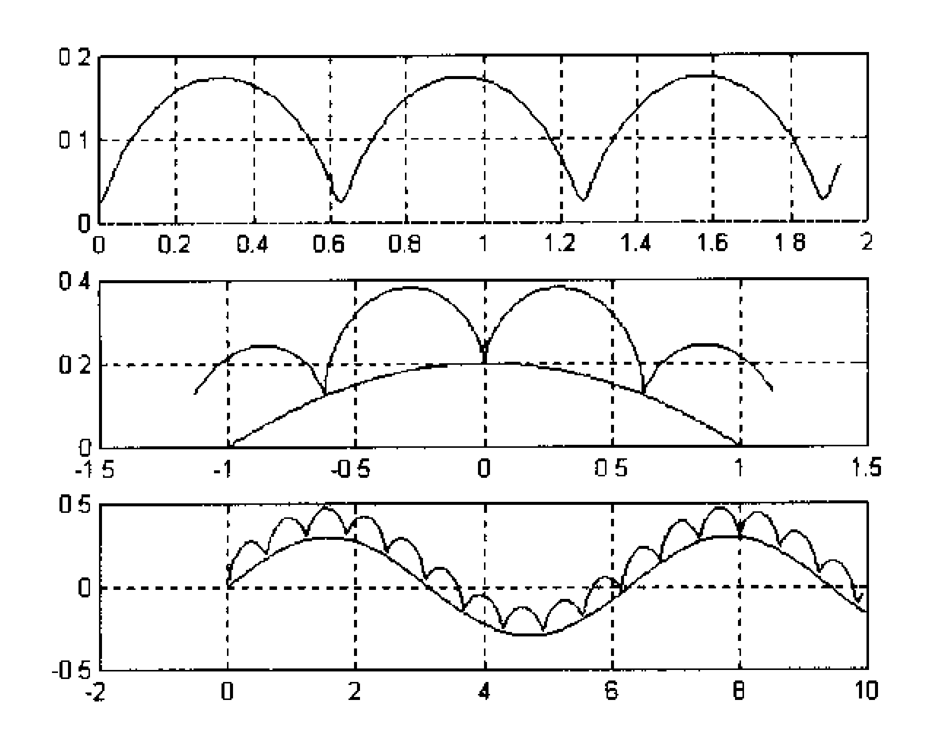


图 10.11 自行车饰物运行曲线

例 10.5 广告费用与效应。某装饰材料公司欲以每桶 2 元的价钱购进一批彩漆。一般来说,随着彩漆售价的提高,预期销售量将减少,并对此进行了估算,见表 10.1。

表 10.1 售价与预期销售量

售价(元)	预期销售量(桶)	售价(元)	预期销售量(桶)
2.00	41 000	2.50	38 000
3.00	34 000	3.50	32 000
4.00	29 000	4.50	28 000
5.00	25 000	5.50	22 000
6.00	20 000		

为了尽快收回资金并获得较多的赢利,装饰材料公司打算做广告。投入一定的广告费后,销售量将有一个增长,可由销售增长因子来表示。例如,投入 4 万元的广告费,销售增长因子为 1.95,即销售将是预期销售量的 1.95 倍。根据经验,广告费与销售增长因子的关系见表 10.2。

表 10.2　广告费与销售增长因子

广告费(元)	销售增长因子	广告费(元)	销售增长因子
0	1.00	10 000	1.40
20 000	1.70	30 000	1.85
40 000	1.95	50 000	2.00
60 000	1.95	70 000	1.80

现在的问题是装饰材料公司采取怎样的营销战略使得预期的利润最大?

设 x 表示售价(单位:元),y 表示预期销售量(单位:桶),z 表示广告费(单位:元),k 表示销售增长因子。投入广告费后,实际销售量记为 s,获得的利润记为 P(单位:元)。由表 10.1 易见,预期销售量 y 随着售价 x 的增加而单调下降,而销售增长因子 k 在开始时随着广告费 z 的增加而增加,在广告费 z 等于 50 000 元时达到最大值,然后在广告费增加时反而有所回落,为此可用 MATLAB 画出散点图,用程序实现如下:

```
x1 = 2:0.5:6;
y1 = [41000,38000,34000,32000,29000,28000,25000,22000,20000];
x2 = 0:10000:70000;
y2 = [1.0,1.4,1.7,1.85,1.95,2.00,1.95,1.8];
subplot(2,1,1);plot(x1,y1,' o ');title('售价与预期销售量');
subplot(2,1,2);plot(x2,y2,' o ');title('广告费与销售增长因子');
```

程序执行后,得到图 10.12。

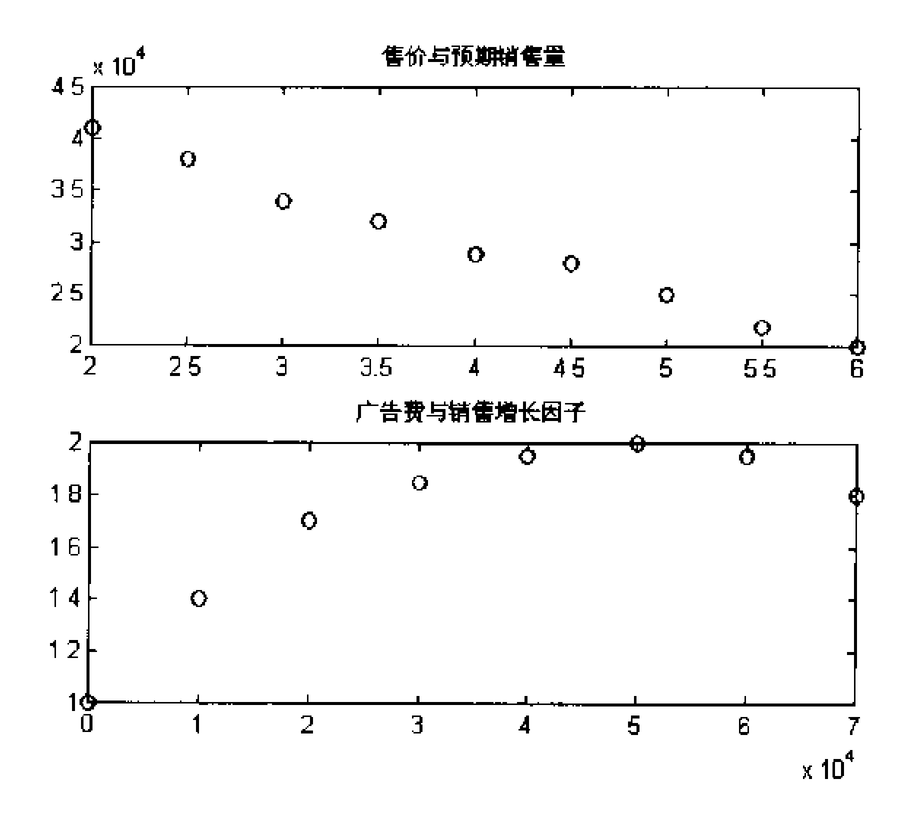


图 10.12　销售关系图

从图 10.12 易见，售价 x 与预期销售量 y 近似于一条直线，广告费 z 与销售增长因子 k 近似于一条二次曲线。为此可令

$$y = a + bx,\ k = c + dz + ez^2$$

其中系数 a,b,c,d,e 是待定参数。

投入广告费后，实际销售量 s 等于预期销售量 y 乘以销售增长因子 k，即 $s = ky$。所获得的利润为

$$\begin{aligned} P &= \text{收入} - \text{支出} = \text{销售收入} - \text{成本} - \text{广告费} \\ &= sx - 2s - z = kxy - 2ky - z = ky(x-2) - z = (c + dz + ez^2)(a + bx)(x-2) - z \end{aligned}$$

我们期望利润 P 达到最大，即

$$\max\ P = (c + dz + ez^2)(a + bx)(x-2) - z,\text{其中 } x>0, z>0$$

首先利用 MATLAB 编程计算参数 a,b,c,d,e。

```
format long;
x1 = 2:0.5:6;
y1 = [41000,38000,34000,32000,29000,28000,25000,22000,20000];
x2 = 0:10000:70000;
y2 = [1.0,1.4,1.7,1.85,1.95,2.00,1.95,1.8];
a1 = polyfit(x1,y1,1);
a2 = polyfit(x2,y2,2);
disp(a1),disp(a2)
```

运行结果为：

```
1.0e+004 *
-0.51333333333333   5.04222222222222
-0.00000000042560   0.00004092261905   1.01875000000000
```

即拟合曲线分别为

$$y = 50\,422.2 - 5\,133.33x$$

$$k = 1.018\,75 + 4.092\,26 \times 10^{-5} z - 4.256\,0 \times 10^{-10} z^2$$

其次用 MATLAB 求解优化模型。因 MATLAB 中仅能求极小值，为此将优化模型转化为

$$\min(-P) = z - (c + dz + ez^2)(a + bx)(x-2),\text{其中 } x>0, z>0$$

建立函数文件 p.m：

```
function f = p(x)
f = x(2) - (1.01875 + 4.09226e-5 * x(2) - 4.2560e-10 * x(2)^2) * (50422.2 - 5133.33 * x(1)) * (x(1) - 2)
```

利用 fmins 寻优：

```
format short e
x = fmins('p',[0,0])        %寻优从坐标原点开始
```

运行结果为

```
f =
  -1.1666e+005
x =
   5.9113e+000   3.3116e+004
```

由此可见：$x=5.9113$，$z=33116$，函数 p 达到最大值 116 660。代入公式，得 $y=20078$，将 c,d,e,z 的值代入公式，得 $k=1.9072$。从上面的计算可知，投入 33 116 元的广告费后，实际销售量为 $ky=38292$，利润为 116 660 元。

10.4　MATLAB 在工程结构分析中的应用

10.4.1　概述

工程结构包括土木工程结构和机械工程结构等。工程结构分析主要的根据是力学原理。经典力学原理基本上沿着两条路线进行。一条是基于牛顿运动定律，在静力分析中，主要遵循力的平衡原理，加上组成结构材料的本构关系和应变、位移的几何协调关系可以导出微分方程。另一条是基于功、能原理，它以能量原理（如最小势能原理，虚位移原理等）为基础，可以导出需要求解的积分方程。

不管是解微分方程还是解积分方程，均需求出函数 $y=f(x)$，使之满足方程并在边界上满足边界条件。对于简单的问题可以求得其解析解，但工程实际问题是复杂的，往往很难求得其实用的解析解，因此，应用计算机得到其数值解成了可行的解决问题的途径。常用的数值方法有：差分法、有限元法、加权残值法、边界元法等，这些解法通常都有大量的矩阵运算以及其他数值计算。MATLAB 具有强大的科学计算功能，这使得人们可以用它来代替 FORTRAN 等传统的编程语言。在计算要求相同的情况下，使用 MATLAB 编程，工作量会大大减少。例如在结构动力学中利用 FORTRAN 求解结构的自由振动，需调用 Jacobi 子程序求解矩阵的特征值及对应的特征向量，而且要求该矩阵必须为实对称矩阵，程序繁杂，且对一般用户来说，要看懂程序算法实属不易，而采用 MATLAB 编制该自由振动的于程序时只需调用两个函数：求逆矩阵的 inv 函数及求特征值、特征向量的 eig 函数。下面简单地介绍 MATLAB 在工程结构分析中的应用。

10.4.2　实例

例 10.6　简支梁荷载情况如图 10.13 所示。求其弯矩、转角和挠度。已知 $L=8$ m，$q=500$ N/m，$F_0=1000$ N，$M_0=800$ N·m，$E=200\times10^9$ N/m，$I=2\times10^{-6}$ m^4。

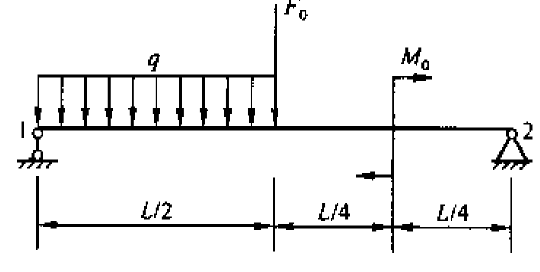


图 10.13　简支梁受力图

从材料力学的知识可知道，由弯矩求转角要经过一次不定积分，而由转角求挠度又要经过一次不定积分，通常这是很麻烦而且容易出错的，而在 MATLAB 中，可用 cumsum 函数或 cumtrapz 函数作近似的不定积分，只要 x 取得足够密，其结果将相当准确，且程序非常简单。

由平衡方程可求出支撑反力 N_1 和 N_2

$$N_1=\left(\frac{qL}{2}\cdot\frac{3L}{4}+F_0\cdot\frac{L}{2}-M_0\right)\Big/ L,\quad N_2=\frac{qL}{2}+F_0-N_1$$

故各段弯矩方程为

$$M_1 = N_1 x - q \cdot \frac{x^2}{2}, \qquad 0 \leqslant x \leqslant L/2$$

$$M_2 = N_2(L - x) - M_0, \quad L/2 \leqslant x \leqslant \frac{3L}{4}$$

$$M_3 = N_2(L - x), \qquad \frac{3L}{4} \leqslant x \leqslant L$$

对 M/EI 作积分,得转角 A,再作一次积分,得到挠度 Y,每次积分都要出现一个待定积分常数

$$A = \int_0^x \frac{M}{EI} dx + C_1 = A_0(x) + C_1$$

此处设 $A_0(x)$ = cumtrapz(M) * dx/EI

$$Y = \int_0^x A dx + C_2 = \int_0^x A_0(x) dx + C_1 x + C_2 = Y_0(x) + C_1 x + C_2$$

此处设 $Y_0(x)$ = cumtrapz(A_0) * dx。

两个待定积分常数 C_1 和 C_2 可由边界条件 $Y(0) = 0$ 及 $Y(L) = 0$ 确定

$$Y(0) = Y_0(0) + C_2 = 0$$

$$Y(L) = Y_0(L) + C_1 L + C_2 = 0$$

于是可得

$$\begin{bmatrix} 0 & 1 \\ L & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} -Y_0(0) \\ -Y_0(L) \end{bmatrix}$$

由此编写 MATLAB 程序如下:

```
L = 8;q = 500;F0 = 1000;M0 = 800;E = 200e9;I = 2e - 6;
N1 = (3 * q * L^2/8 + F0 * L/2 - M0)/L;N2 = (q * L^2/8 + F0 * L/2 + M0)/L;
x = linspace(0,L,101);dx = L/100;
M1 = N1 * x(1:51) - q * x(1:51).^2/2;                 %分 3 段用数组列出 M 的表达式
M2 = N2 * (L - x(52:76)) - M0;
M3 = N2 * (L - x(77:101));
M = [M1,M2,M3];
A0 = cumtrapz(M) * dx/(E * I);                        %由 M 积分求转角(未记积分常数)
Y0 = cumtrapz(A0) * dx;                               %由转角积分求挠度(未记积分常数)
C = [0,1;L,1] \ [ - Y0(1); - Y0(101)];                %由边界条件求积分常数
A = A0 + C(1);Y = Y0 + C(1) * x + C(2);               %求转角和挠度的完整值
subplot(3,1,1);plot(x,M);grid on;
subplot(3,1,2);plot(x,A);grid on;
subplot(3,1,3);plot(x,Y);grid on;
```

运行程序后得到图 10.14 所示的弯矩、转角和挠度曲线。

梯形积分累加函数 cumtrapz 与梯形积分函数 trapz 的不同在于它逐点给出积分值,因而得出一个积分序列,而 trapz 只给出积分到终点的一个值。

例 10.7　计算图 10.15 所示直梁的自振频率(基频)。已知 $E = 2 \times 10^{11}$ Pa,$\rho = 7\ 860$ kg/m^3,$A =$

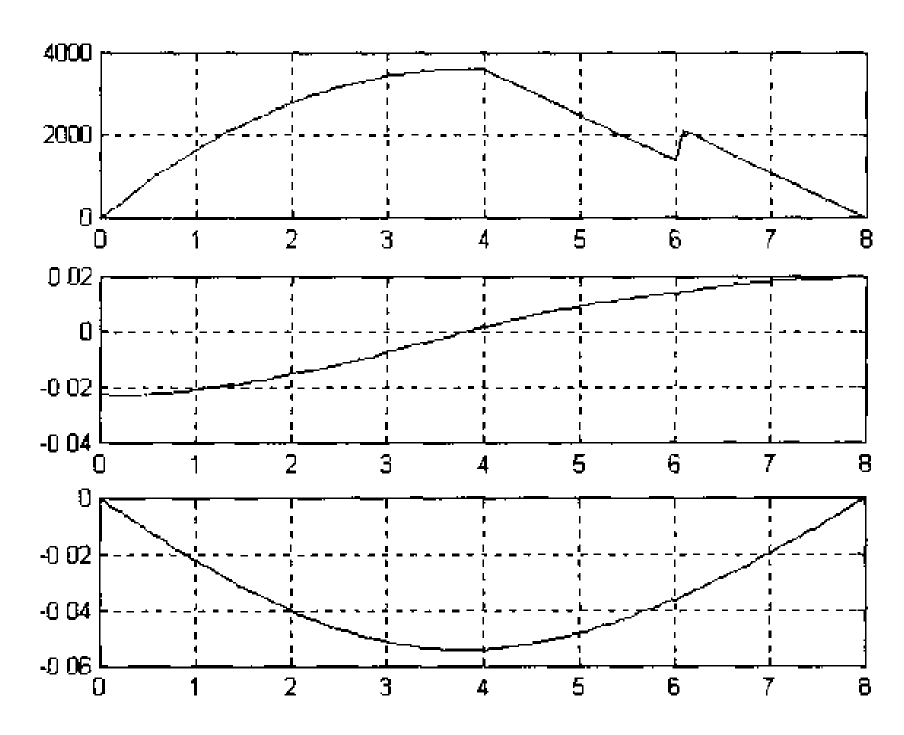


图 10.14 弯矩、转角和挠度曲线

$1.14\times10^{-3}\ \mathrm{m}^2$，$I=1.03\times10^{-7}\ \mathrm{m}^4$。

在结构的自由振动分析中，均不计入阻尼力的作用，其运动方程为

$$[\boldsymbol{M}]\{\ddot{\delta}\}+[\boldsymbol{K}]\{\delta\}=0$$

设各点作简谐振动，即

$$\{\delta\}=\{\delta_0\}\sin\omega t$$

$$\{\ddot{\delta}\}=-\omega^2\{\delta_0\}\sin\omega t$$

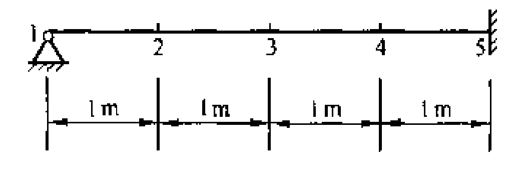


图 10.15 直梁的自由振动

代入上式得运动方程

$$([\boldsymbol{K}]-\omega^2[\boldsymbol{M}])\{\delta_0\}=0$$

在自由振动时，结构各结点的振幅值$\{\delta_0\}$不可能全等于零，要使上式（也就是齐次方程组）有不全为零的解，则其系数行列式应等于零，即

$$|[\boldsymbol{K}]-\omega^2[\boldsymbol{M}]|=0$$

这便是结构自由振动的频率方程，展开后是关于 ω 的高次方程，方程的次数与矩阵$[\boldsymbol{K}]$和$[\boldsymbol{M}]$的阶数相同。由频率方程解出的根 ω_1、ω_2、ω_3、…，分别称为第一、第二、第三、…自振频率。将运动方程左乘$[\boldsymbol{M}]^{-1}$，则

$$([\boldsymbol{M}]^{-1}[\boldsymbol{K}]-\omega^2[\boldsymbol{I}])\{\delta_0\}=0$$

这时求解自振频率也就是求矩阵$[\boldsymbol{M}]^{-1}[\boldsymbol{K}]$的特征值。下面就如何形成$[\boldsymbol{K}]$矩阵和$[\boldsymbol{M}]$矩阵加以说明。

(1) 整体刚度矩阵$[\boldsymbol{K}]$的形成

设有单跨梁如图 10.16 所示，两端结点编号为 1，2。由结构力学可知，杆端力（指剪力和弯矩）与杆端位移（指竖向位移和角位移）的关系，即单元刚度矩阵可表达为

$$[\boldsymbol{K}]_e=\begin{bmatrix} \frac{12EI}{l^3} & -\frac{6EI}{l^2} & -\frac{12EI}{l^3} & -\frac{6EI}{l^2} \\ -\frac{6EI}{l^2} & \frac{4EI}{l} & \frac{6EI}{l^2} & \frac{2EI}{l} \\ -\frac{12EI}{l^3} & \frac{6EI}{l^2} & \frac{12EI}{l^3} & \frac{6EI}{l^2} \\ -\frac{6EI}{l^2} & \frac{2EI}{l} & \frac{6EI}{l^2} & \frac{4EI}{l} \end{bmatrix}$$

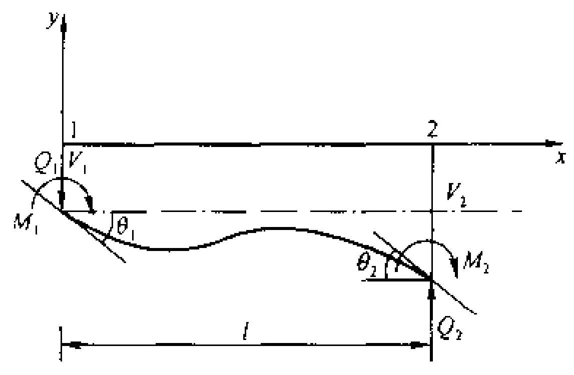


图 10.16 单跨梁示意图

在梁的分析中，忽略梁的轴向变形，不计梁的轴向位移，因而每一个结点只有 2 个位移自由度。一个杆件有 2 个结点，共有 4 个位移自由度，所以单元刚度矩阵是 4×4 阶的。为了集合整体刚度矩阵，需要找出同一结点位移在单元刚度矩阵中的位置与在整体刚度矩阵中的位置之间的对应关系。根据这种对应关系，利用 MATLAB 语言很容易实现。

(2) 整体质量矩阵[$\boldsymbol{M}$]的形成

如图 10.16 所示梁的单元质量矩阵为

$$[\boldsymbol{M}]_e=\frac{\rho Al}{420}\begin{bmatrix} 156 & -22l & 54 & 13l \\ -22l & 4l^2 & -13l & -3l^2 \\ 54 & -13l & 156 & 22l \\ 13l & -3l^2 & 22l & 4l^2 \end{bmatrix}$$

质量矩阵的集合方法与刚度矩阵的集合方法一样。

(3) 支承约束条件的处理

刚度矩阵和质量矩阵集合完成后引入支承条件，采用“缩减法”将被约束的位移自由度缩减掉，即将[$\boldsymbol{K}$]和[$\boldsymbol{M}$]矩阵中与支承处相对应的行和列全部划掉。在自由振动中，结构自振频率数等于结构中能自由位移的自由度数。

将图 10.15 中的梁划分为 4 段，共 5 个结点 10 个自由度(其中 1 结点的竖向位移与 5 结点的竖向及转动位移受约束，即矩阵为 7×7 矩阵)，编制程序如下：

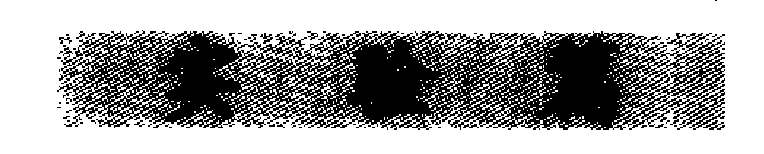
```
E = 2e11;I = 1.03e-7;A = 1.14e-3;rho = 7860;L = 1;
X = [12/L/L/L, -6/L/L, -12/L/L/L, -6/L/L];
Ke = E * I * [X; -6/L/L,4/L,6/L/L,2/L; -X; -6/L/L,2/L,6/L/L,4/L];
K = Ke;KK = Ke;
for i = 2:4            %生成整体刚度矩阵
    K = [K,zeros(2 * i,1),zeros(2 * i,1)];
    K = [K;zeros(1,2 * i + 2);zeros(1,2 * i + 2)];
    KK = [zeros(2 * i,1),zeros(2 * i,1),KK];
    KK = [zeros(1,2 * i + 2);zeros(1,2 * i + 2);KK];
    K = K + KK;
end
Me = [156, -22 * L,54,13 * L; -22 * L,4 * L * L, -13 * L, -3 * L * L;54,…
    -13 * L,156,22 * L;13 * L, -3 * L * L,22 * L,4 * L * L];
Me = rho * A * L/420 * Me;
```

```
M = Me; MM = Me;
for i = 2:4             %生成整体质量矩阵
    M = [M,zeros(2 * i,1),zeros(2 * i,1)];
    M = [M;zeros(1,2 * i + 2);zeros(1,2 * i + 2)];
    MM = [zeros(2 * i,1),zeros(2 * i,1),MM];
    MM = [zeros(1,2 * i + 2);zeros(1,2 * i + 2);MM];
    M = M + MM;
end
K(1,:) = [];K(:,1) = [];K(8,:) = [];K(:,8) = [];K(8,:) = [];K(:,8) = [];        %处理约束条件
M(1,:) = [];M(:,1) = [];M(8,:) = [];M(:,8) = [];M(8,:) = [];M(:,8) = [];
AA = inv(M) * K;
W = eig(AA);                            %求特征值
W = sqrt(W);
disp(W ');
```

程序运行结果为：

```
1.0e + 003  *
    2.2607    1.5615    0.9780    0.5999    0.3194    0.1507    0.0462
```

即直梁自由振动基频为 46.2。



实验要求

学习程序设计,上机实验是十分重要的环节。为了方便读者上机练习,在实验篇设计了15个实验。这些实验和课堂教学紧密配合,通过有针对性的上机实验,可以更好地熟悉MATLAB的功能,掌握MATLAB程序设计的方法,并培养一定的应用开发能力。每个实验安排2机时左右。读者也可以根据实际情况从每个实验中选择部分内容作为上机练习。另外,基础篇和应用篇各章后面的部分习题也可以作为实验内容的补充。

为了达到理想的实验效果,读者务必做到:

(1) 实验前认真准备,要根据实验目的和实验内容,复习好实验中可能要用到的命令,想好编程的思路,做到胸有成竹,提高上机效率。

(2) 实验过程中积极思考,要深入分析命令、程序的执行结果以及各种屏幕信息的含义、出现的原因并提出解决办法。

(3) 实验后认真总结,要总结本次实验有哪些收获,还存在哪些问题,并写出实验报告。实验报告应包括实验目的、实验内容、流程图、程序(命令)清单、运行结果以及实验的收获与体会等内容。

程序设计和应用开发能力的提高需要不断的上机实践和长期的积累,在上机过程中会碰到各种各样的问题,分析问题和解决问题的过程就是经验积累的过程。只要读者按照上面3点要求去做,在学完本课程后就一定会有很大的收获,计算机应用能力就会有很大提高。

实验一 MATLAB运算基础

一、实验目的

1. 熟悉启动和退出MATLAB的方法。
2. 熟悉MATLAB命令窗口的组成。
3. 掌握建立矩阵的方法。
4. 掌握MATLAB各种表达式的书写规则以及常用函数的使用。

二、实验内容

1. 先求下列表达式的值,然后显示MATLAB工作空间的使用情况并保存全部变量。

(1) $z^1 = \dfrac{2\sin 85°}{1+e^2}$

(2) $z2 = \dfrac{1}{2}\ln\left(x+\sqrt{1+x^2}\right)$,其中 $x = \begin{bmatrix} 2 & 1+2i \\ -0.45 & 5 \end{bmatrix}$

(3) $z3=\frac{e^{0.3a}-e^{0.3a}}{2}\cdot\sin(a+0.3),a=-3.0,-2.9,-2.8,\cdots,2.8,2.9,3.0$

提示：利用冒号表达式生成 a 向量，求各点的函数值时用点乘运算。

(4) $z4=\begin{cases}t^2, & 0\leq t<1\\ t^2-1, & 1\leq t<2\\ t^2-2t+1, & 2\leq t<3\end{cases}$，其中 $t=0:0.5:2.5$

提示：用逻辑表达式求分段函数值。

2. 已知

$$A=\begin{bmatrix}12 & 34 & -4\\ 34 & 7 & 87\\ 3 & 65 & 7\end{bmatrix},\quad B=\begin{bmatrix}1 & 3 & -1\\ 2 & 0 & 3\\ 3 & -2 & 7\end{bmatrix}$$

求下列表达式的值：

(1) A+6*B 和 A-B+I(其中 I 为单位矩阵)。

(2) A*B 和 A.*B。

(3) A^3 和 A.^3。

(4) A/B 及 B\A。

(5) [A,B]和[A([1,3],:);B^2]。

3. 设有矩阵 A 和 B

$$A=\begin{bmatrix}1 & 2 & 3 & 4 & 5\\ 6 & 7 & 8 & 9 & 10\\ 11 & 12 & 13 & 14 & 15\\ 16 & 17 & 18 & 19 & 20\\ 21 & 22 & 23 & 24 & 25\end{bmatrix},\quad B=\begin{bmatrix}3 & 0 & 16\\ 17 & -6 & 9\\ 0 & 23 & -4\\ 9 & 7 & 0\\ 4 & 13 & 11\end{bmatrix}$$

(1) 求它们的乘积 C。

(2) 将矩阵 C 的右下角 3×2 子矩阵赋给 D。

(3) 查看 MATLAB 工作空间的使用情况。

4. 完成下列操作：

(1) 求[100,999]之间能被 21 整除的数的个数。

提示：先利用冒号表达式，再利用 find 和 length 函数。

(2) 建立一个字符串向量，删除其中的大写字母。

提示：利用 find 函数和空矩阵。

实验二　选择结构程序设计

一、实验目的

1. 掌握建立和执行 M 文件的方法。
2. 掌握利用 if 语句实现选择结构的方法。
3. 掌握利用 switch 语句实现多分支选择结构的方法。
4. 掌握 try 语句的使用。

二、实验内容

1. 求下列分段函数的值。

$$y=\begin{cases}x^2+x-6, & x<0\text{ 且 }x\neq -3\\ x^2-5x+6, & 0\leqslant x<10, x\neq 2\text{ 且 }x\neq 3\\ x^2-x-1, & \text{其他}\end{cases}$$

要求：

(1) 用 if 语句实现，分别输出 $x=-5.0,-3.0,1.0,2.0,2.5,3.0,5.0$ 时的 y 值。

提示：x 的值从键盘输入，可以是向量。

(2) 仿照实验一第 1 题第 4 小题，用逻辑表达式实现，从而体会 MATLAB 逻辑表达式的一种应用。

2. 输入一个百分制成绩，要求输出成绩等级 A,B,C,D,E。其中 90 ~ 100 分为 A，80 ~ 89 分为 B，70 ~ 79 分为 C，60 ~ 69 分为 D，60 分以下为 E。

要求：

(1) 分别用 if 语句和 switch 语句实现。

(2) 输入百分制成绩后要判断该成绩的合理性，对不合理的成绩应输出出错信息。

3. 假定某地区电话收费标准为：通话时间在 3 分钟以下，收费 0.50 元；3 分钟以上，则每超过 1 分钟加收 0.15 元；在 7:00 ~ 22:00 之间通话者，按上述收费标准全价收费，在其他时间通话者，按上述收费标准半价收费。计算某人在 $t1$ 时间通话至 $t2$ 时间，应缴多少电话费。

提示：

(1) $t1$、$t2$ 从键盘输入，通话时间为 $t2$ 减去 $t1$，相减时可以将 $t1$、$t2$ 化成以秒为单位再相减。

(2) 为了简化程序，根据开始通话的时间来判断是否享受半价收费。

(3) 也可以用 clock 函数得到机器时间，可利用帮助功能查询该函数的用法。以程序开始运行时的时间作为开始通话的时间 $t1$，程序设暂停语句，以暂停结束的时间作为结束通话的时间 $t2$。

4. 建立 5×6 矩阵，要求输出矩阵第 n 行元素。当 n 值超过矩阵的行数时，自动转为输出矩阵最后一行元素，并给出出错信息。

实验三 循环结构程序设计

一、实验目的

1. 掌握利用 for 语句实现循环结构的方法。
2. 掌握利用 while 语句实现循环结构的方法。
3. 熟悉利用向量运算来代替循环操作的方法。

二、实验内容

1. 根据 $\frac{\pi^2}{6}=\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}$，求 π 的近似值。当 n 分别取 100、1 000、10 000 时，结果是多少？

要求：分别用循环结构和向量运算（使用 sum 函数）来实现。

2. 根据 $y=1+\frac{1}{3}+\frac{1}{5}+\cdots+\frac{1}{2n-1}$，求：

(1) $y<3$ 时的最大 n 值。

(2) 与(1)的 n 值对应的 y 值。

3. 一个三位整数各位数字的立方和等于该数本身则称该数为水仙花数。试输出全部水仙花数。

要求：

(1) 用循环结构实现。

(2) 用向量运算来实现。

提示：全部 3 位整数组成向量 M；分别求 M 各元素的个位、十位、百位数字，组成向量 M1、M2、M3；向量 N = M1 .* M1 .* M1 + M2 .* M2 .* M2 + M3 .* M3 .* M3；向量 K = M - N；显然 K 中 0 元素的序号即 M 中水仙花数的序号。

4. 已知

$$\begin{cases} f_1=1 \\ f_2=0 \\ f_3=1 \\ f_n=f_{n-1}-2f_{n-2}+f_{n-3}, n>3 \end{cases}$$

求 $f_1\sim f_{100}$ 中：

(1) 最大值、最小值、各数之和。

(2) 正数、零、负数的个数。

提示：可以考虑使用 MATLAB 有关函数来实现。

实验四　函数与文件

一、实验目的

1. 掌握定义和调用 MATLAB 函数的方法。
2. 掌握 MATLAB 文件的基本操作。

二、实验内容

1. 定义一个函数文件,求给定复数的指数、对数、正弦和余弦,并在命令文件中调用该函数文件。

2. 一物理系统可用下列方程组来表示

$$\begin{bmatrix} m_1\cos\theta & -m_1 & -\sin\theta & 0 \\ m_1\sin\theta & 0 & \cos\theta & 0 \\ 0 & m_2 & -\sin\theta & 0 \\ 0 & 0 & -\cos\theta & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ N_1 \\ N_2 \end{bmatrix} = \begin{bmatrix} 0 \\ m_1 g \\ 0 \\ m_2 g \end{bmatrix}$$

从键盘输入 m_1、m_2 和 θ 的值,求 a_1、a_2、N_1 和 N_2 的值。其中 g 取 9.8,输入 θ 时以角度为单位。

要求:定义一个求线性方程组 $\boldsymbol{AX}=\boldsymbol{B}$ 根的函数文件,然后在命令文件中调用该函数文件。

3. 一个自然数是素数,且它的各位数字位置经过任意对换后仍为素数,则称是绝对素数。例如 13 是绝对素数。试求所有两位的绝对素数。

要求:定义一个判断素数的函数文件。

4. 统计一个文本文件中每个英文字母出现的次数,不区分字母的大小写。

实验五　高层绘图操作

一、实验目的

1. 掌握绘制二维图形的常用函数。
2. 掌握绘制三维图形的常用函数。
3. 掌握绘制图形的辅助操作。

二、实验内容

1. 已知 $y1=x^2$, $y2=\cos(2x)$, $y3=y1*y2$,完成下列操作:

(1) 在同一坐标系下用不同的颜色和线型绘制3条曲线。

(2) 以子图形式绘制3条曲线。

(3) 分别用条形图、阶梯图、杆图和填充图绘制3条曲线。

2. 绘制极坐标曲线 $\rho = a\sin(b+n\theta)$,并分析参数 a,b,n 对曲线形状的影响。

3. 分别用 plot 和 fplot 函数绘制函数 $y=\sin\frac{1}{x}$ 的曲线,分析两曲线的差别。

4. 绘制函数的曲面图和等高线。

(1) $z=(x^2-2x)e^{-x^2-y^2-xy}$

(2) $f(x,y)=\frac{1}{\sqrt{(x-1)^2+y^2}}-\frac{1}{\sqrt{(x+1)^2+y^2}}$

提示:绘制三维曲面图,首先要选定一平面区域并在该区域产生网格坐标矩阵。在做本题前,先分析并上机验证下列命令的执行结果,从中体会产生网格坐标矩阵的方法。

```
[x,y] = meshgrid(-1:0.5:2,1:5)
```

实验六 低层绘图操作

一、实验目的

1. 掌握图形对象属性的基本操作。

2. 掌握利用图形对象进行绘图操作的方法。

二、实验内容

1. 建立一个图形窗口,使之背景颜色为红色,并在窗口上保留原有的菜单项,而且在按下鼠标器的左键之后显示出 Left Button Pressed 字样。

2. 先利用缺省属性绘制曲线 $y=x^2e^{2x}$,然后通过图形句柄操作来改变曲线的颜色、线型和线宽,并利用文字对象给曲线添加文字标注 $y=x^2e^{2x}$。

3. 利用曲面对象绘制曲面 $v(x,t)=10e^{-0.01x}\sin(2000\pi t-0.2x+\pi)$,并要求分别绘制曲面在 $x-y$、$x-z$ 和 $y-z$ 平面上的投影。

提示:通过视点的设置来绘制出曲面在两两平面上的投影。

4. 以任意位置子图形式绘制出正弦、余弦、正切和余切函数曲线。

提示:利用坐标轴对象对图形窗口做任意分割。

实验七　线性代数中的数值计算问题

一、实验目的

1. 掌握生成特殊矩阵的方法。
2. 掌握矩阵分析的方法。
3. 掌握线性方程组的求解方法。

二、实验内容

1. 产生5阶希尔伯特矩阵 $\boldsymbol{H}$ 和5阶帕斯卡矩阵 $\boldsymbol{P}$，且求其行列式的值 Hh 和 Hp 以及它们的条件数 Th 和 Tp，判断哪个矩阵性能更好，为什么？

2. 已知

$$\boldsymbol{A}=\begin{bmatrix}-29 & 6 & 18\\ 20 & 5 & 12\\ -8 & 8 & 5\end{bmatrix}$$

求 $\boldsymbol{A}$ 的特征值及特征向量，并分析其数学意义。

3. 分别用3种不同的数值方法解线性方程组。

$$\begin{cases}6x+5y-2z+5u=-4\\ 9x-y+4z-u=13\\ 3x+4y+2z-2u=1\\ 3x-9y+2u=11\end{cases}$$

4. 不用rot90函数，实现方阵左旋90°或右旋90°的功能。例如，原矩阵为 $\boldsymbol{A}$，$\boldsymbol{A}$ 左旋后得到 $\boldsymbol{B}$，右旋后得到 $\boldsymbol{C}$

$$\boldsymbol{A}=\begin{bmatrix}1 & 4 & 7 & 10\\ 2 & 5 & 8 & 11\\ 3 & 6 & 9 & 12\end{bmatrix},\boldsymbol{B}=\begin{bmatrix}10 & 11 & 12\\ 7 & 8 & 9\\ 4 & 5 & 6\\ 1 & 2 & 3\end{bmatrix},\boldsymbol{C}=\begin{bmatrix}3 & 2 & 1\\ 6 & 5 & 4\\ 9 & 8 & 7\\ 12 & 11 & 10\end{bmatrix}$$

要求：定义一个函数文件，然后在命令文件中调用该函数文件。函数文件的调用格式为myrot(A,t)，其中 $\boldsymbol{A}$ 在调用前为原矩阵，调用后为经旋转后的矩阵，t 确定是左旋还是右旋，当 $t=1$ 时为左旋，$t=-1$ 时为右旋，t 为其他值时显示出错信息。

提示：先将 $\boldsymbol{A}$ 转置，再作上下翻转，则完成左旋90°；若将 $\boldsymbol{A}$ 转置后作左右翻转，则完成右旋90°。

实验八　数据处理和多项式计算

一、实验目的

1. 掌握数据统计和分析的方法。
2. 掌握数值插值与曲线拟合的方法及其应用。
3. 掌握多项式的常用运算。

二、实验内容

1. 将100个学生5门功课的成绩存入矩阵 **P** 中，进行如下处理：

(1) 分别求每门课的最高分、最低分及相应学生序号。

(2) 分别求每门课的平均分和标准方差。

(3) 5门课总分的最高分、最低分及相应学生序号。

(4) 将5门课总分按从大到小顺序存入 zcj 中，相应学生序号存入 xsxh。

提示：上机调试时，为避免输入学生成绩的麻烦，可用取值范围在[45,95]之间的随机矩阵来表示学生成绩。

2. 某气象观测站测得某日 6:00 ~ 18:00 之间每隔 2 h 的室内外温度(℃)如表1所示。

表1　室内外温度观测结果(℃)

时间 h	6	8	10	12	14	16	18
室内温度 $t1$	18.0	20.0	22.0	25.0	30.0	28.0	24.0
室外温度 $t2$	15.0	19.0	24.0	28.0	34.0	32.0	30.0

试用三次样条插值分别求出该日室内外 6:30 ~ 17:30 时之间每隔 2 h 各点的近似温度(℃)。

3. 已知 $\lg(x)$ 在[1,101]区间11个整数采样点的函数值如表2所示。

表2　$\lg(x)$在10个采样点的函数值

x	1	11	21	31	41	51	61	71	81	91	101
$\lg(x)$	0	1.041 4	1.322 2	1.491 4	1.612 8	1.707 6	1.785 3	1.851 3	1.908 5	1.959 0	2.004 3

试求 $\lg(x)$ 的5次拟合多项式 $p(x)$，并分别绘制出 $\lg(x)$ 和 $p(x)$ 在[1,101]区间的函数曲线。

4. 有3个多项式 $P_1(x) = x^4 + 2x^3 + 4x^2 + 5$，$P_2(x) = x + 2$，$P_3(x) = x^2 + 2x + 3$，试进行下列操作：

(1) 求 $P(x) = P_1(x) + P_2(x) * P_3(x)$。

(2) 求 P(x)的根。

(3) 当 x 取矩阵A 的每一元素时,求 $P(x)$的值。其中

$$A=\begin{bmatrix} -1 & 1.2 & -1.4 \\ 0.75 & 2 & 3.5 \\ 0 & 5 & 2.5 \end{bmatrix}$$

(4) 当以矩阵 A 为自变量时,求 $P(x)$的值。其中 A 的值与(3)相同。

实验九　数值微积分与方程数值求解

一、实验目的

1. 掌握求数值导数和数值积分的方法。
2. 掌握常微分方程数值求解的方法。
3. 掌握非线性代数方程数值求解的方法。

二、实验内容

1. 求函数在指定点的数值导数。

$$f(x)=\begin{vmatrix} x & x^2 & x^3 \\ 1 & 2x & 3x^2 \\ 0 & 2 & 6x \end{vmatrix},\quad x=1,2,3$$

2. 用数值方法求定积分。

(1) $I_1=\int_0^{2\pi}\sqrt{\cos t^2+4\sin (2t)^2+1}\,\mathrm{d}t$ 的近似值。

(2) $I_2=\int_0^1\frac{\ln (1+x)}{1+x^2}\,\mathrm{d}x$

3. 求微分方程的数值解。

$$\begin{cases} \dfrac{x\mathrm{d}^2y}{\mathrm{d}x^2}-5\dfrac{\mathrm{d}y}{\mathrm{d}x}+y=0 \\ y(0)=0 \\ y'(0)=0 \end{cases}$$

4. 求代数方程的数值解。

(1) $3x+\sin x-\mathrm{e}^x=0$ 在 $x_0=1.5$ 附近的根。

(2) 在给定的初值 $x_0=1, y_0=1, z_0=1$ 下,求下列方程组的数值解。

$$\begin{cases} \sin x+y^2+\ln z-7=0 \\ 3x+2y-z^3+1=0 \\ x+y+z-5=0 \end{cases}$$

实验十 符号计算基础与符号微积分

一、实验目的

1. 掌握定义符号对象的方法。
2. 掌握符号表达式的运算法则以及符号矩阵运算。
3. 掌握求符号函数极限及导数的方法。
4. 掌握求符号函数定积分和不定积分的方法。

二、实验内容

1. 已知 $x=6, y=5$,利用符号表达求 $z=\dfrac{x+1}{\sqrt{3+x}-\sqrt{y}}$

提示:定义符号常数 x = sym('6'),y = sym('5')。

2. 已知 $\boldsymbol{P}1=\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}, \boldsymbol{P}2=\begin{bmatrix}1&0&0\\0&1&0\\1&0&1\end{bmatrix}, \boldsymbol{A}=\begin{bmatrix}a&b&c\\d&e&f\\g&h&i\end{bmatrix}$

求:

(1) $\boldsymbol{B}=\boldsymbol{P}1\cdot\boldsymbol{P}2\cdot\boldsymbol{A}$。

(2) $\boldsymbol{B}$ 的逆矩阵并验证结果。

(3) 包括 $\boldsymbol{B}$ 矩阵主对角线元素的下三角阵。

(4) $\boldsymbol{B}$ 的行列式值。

提示:用于数值矩阵分析的有关函数同样适用于符号矩阵。验证结果时须将结果化简。

3. 用符号方法求下列极限或导数。

(1) $\lim\limits_{x\to 0}\dfrac{x(e^{\sin x}+1)-2(e^{\tan x}-1)}{\sin^3 x}$

(2) 已知 $A=\begin{bmatrix}a^x & t^3\\ t\cos x & \ln x\end{bmatrix}$,分别求 $\dfrac{dA}{dx}$、$\dfrac{d^2A}{dt^2}$、$\dfrac{d^2A}{dx\,dt}$

4. 用符号方法求下列积分。

(1) $\displaystyle\int\frac{dx}{1+x^4+x^8}$

(2) $\displaystyle\int_0^{+\infty}\frac{x^2+1}{x^4+1}dx$

实验十一　级数与方程符号求解

一、实验目的

1. 掌握级数求和的方法。
2. 掌握将函数展开为泰勒级数的方法。
3. 掌握微分方程符号求解的方法。
4. 掌握代数方程符号求解的方法。

二、实验内容

1. 级数符号求和。

(1) 计算 $S=\sum_{n=1}^{10}\frac{1}{2n-1}$

(2) 求级数 $\sum_{n=1}^{\infty}n^2x^{n-1}$之和函数，并求 $\sum_{n=1}^{\infty}\frac{n^2}{5^n}$之和。

2. 将 $\ln(x)$在 $x=1$ 处按 5 次多项式展开为泰勒级数。

3. 求微分方程的符号解。

$$\begin{cases}\frac{d^2y}{dx^2}+k^2y=0\\ y(0)=a\\ y'(0)=b\\ a、b、k\text{ 为任意常数}\end{cases}$$

4. 求下列方程和方程组的符号解。

(1) $3xe^x+5\sin x-78.5=0$

(2) $\begin{cases}\sqrt{x^2+y^2}-100=0\\ 3x+5y-8=0\end{cases}$

实验十二　菜 单 设 计

一、实验目的

1. 了解图形用户界面的特点。
2. 掌握菜单设计的方法。

二、实验内容

1. 在图形窗口缺省菜单条上增加一个 Plot 菜单项，利用该菜单项可以在本窗口绘制三维曲面图形。

2. 为图形窗口建立快捷菜单，用以控制窗口的背景颜色和大小。

3. 设计菜单(如图 1 所示)。

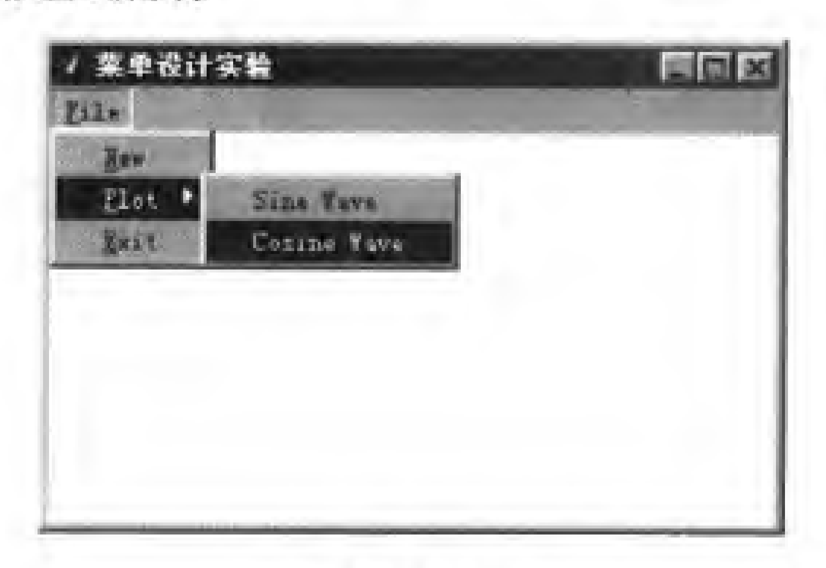


图 1　菜单设计

菜单条仅有 File 菜单项，File 下有 New、Plot 和 Exit 等 3 个选项。选择 New 时利用 Edit 命令建立一个新的 M 文件。选择 Plot 将显示下一级菜单，其中有 Sine Wave 和 Cosine Wave 两个子菜单项，且若选择了其中的 Sine Wave 子菜单项，则将打开一个新的图形窗口并显示出正弦曲线。若选择了其中的 Cosine Wave 子菜单项，则将打开一个新的图形窗口并显示出余弦曲线。如果选择 Exit 菜单项，则将关闭窗口并退出用户系统回到 MATLAB 命令窗口。

实验十三　对话框的设计

一、实验目的

1. 掌握建立控件对象的方法。

2. 掌握对话框设计的方法。

二、实验内容

1. 设计一个对话框，其中有一个编辑框和按钮，当单击按钮时，使编辑框的内容加 5。

2. 采用图形用户界面，从键盘输入参数 a，b，n 的值，考察参数对极坐标曲线 $\rho = a\cos(b +$

$n\theta$)的影响。

3. 编一程序，对给定位数的数据产生校验位。

在数据处理中，关键字段值的正确与否对系统的运行至关重要。例如，在各种考试中，准考证号是识别考生的关键字。为了正确地判断准考证号输入的正确性，将 7 位数的准考证号在其前面加一位校验位，构成有校验位的 8 位准考证号。假定产生校验位的公式为：

$$a = \left(\sum_{i=1}^{7} d_i \times i\right) \bmod 10$$

其中 a 为校验位，d_i 第 i 位的数。

例如，某考生准考证号为 2573458，则校验位为

$$a = (2\times7+5\times6+7\times5+3\times4+4\times3+5\times2+8\times1) \bmod 10 = 1$$

因此，带有校验位的 8 位准考证号为 12573458。

要求采用图形用户界面，程序具体功能如下：

(1) 输入 7 位数的准考证号，单击“生成”按钮，则显示对应带有校验位的 8 位准考证号。当输入非 7 位数时，显示出错信息。

(2) 输入带有校验位的 8 位数的准考证号，单击“校验”按钮，校验输入的准考证号正确与否。同样，当输入非 8 位数时，显示出错信息。

(3) 单击“退出”按钮，退出程序。

实验十四　Simulink 的应用

一、实验目的

1. 熟悉 Simulink 的操作环境并掌握构建系统模型的方法。
2. 掌握 Simulink 中子系统模块的建立与封装技术。
3. 对简单系统所给出的数学模型能转化为系统仿真模型并进行仿真分析。

二、实验内容

1. 构建图 2(a)所示的含有齿隙非线性环节的系统模型并进行仿真。其中齿隙非线性环节的特性如图 2(b)所示。

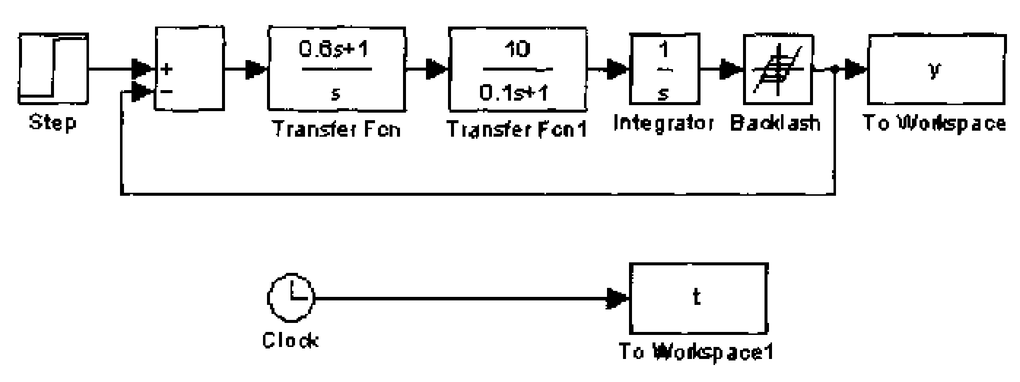
提示：

(1) 齿隙非线性环节位于模块浏览器的非线性模块库(Nonliner)中。

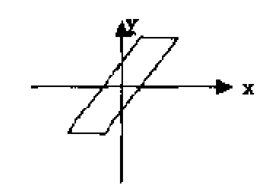
(2) 在仿真之前，对各个的参数进行设置，然后选择 Simulink 菜单中的 Parameters 命令来设置仿真参数(选仿真终止时间为 3 s)，再选择 Simulink 菜单中的 Start 命令即可开始仿真了。

2. PID 控制器是在工业控制中经常用到的模块，在工程应用中其标准的数学模型为

$$U(S) = K_p\left(1 + \frac{1}{T_i S} + \frac{ST_d}{1 + ST_d/N}\right)E(S)$$



(a) 含有非线性环节的控制系统框图



(b) 齿隙非线性环节特性

图 2　典型非线性系统框图

其中采用了一阶环节来近似纯微分动作，故一般选 $N \geq 10$。先建立 PID 控制器模型如图 3 所示，然后建立并封装子系统。

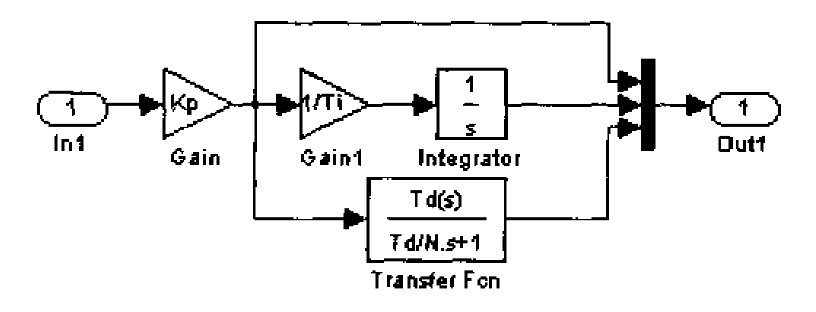


图 3　PID 控制器的 Simulink 描述

提示：

(1) 选中所有模块。选择 Edit 命令下的 Select All 菜单项可以选中所有的模块。

(2) 建立子系统。选择 Edit 命令下的 Create Subsystem 菜单项，则可以由选定的模块和连接关系构造出子系统。

(3) 封装子系统。选择子系统模块，再选择 Edit 命令下的 Mask Subsystem 菜单项，将打开一个对话框，单击该对话框中的 Initialization 标签，打开相应的选项卡，用户可以在该选项卡中输入各个变量的提示 Prompt 和变量名 Variable。

用户还可以打开 Icon 选项卡，在 Drawing commands 栏目下填入 disp(' PID ')字样，这时可以在该图标上显示 PID 字样。

(4) 子系统参数初始化。建立该子系统后，双击该图标，将弹出一个对话框。用户可以在各个栏目上填写默认参数，然后将此子系统存盘备用。

3. 设某简单系统的微分方程为 $X'(t) = -4X(t) + 2u(t)$,式中 $u(t)$是一个幅度为 1,角频率为 1 rad/s 的方波输入信号,试用两种以上的方法建立系统模型并进行仿真,同时利用 Scope 获得 $X(t)$的波形。

实验十五 综合实验

一、实验目的

综合运用所学知识,掌握利用 MATLAB 解决实际问题的方法。

二、实验内容

1. 给出迭代方程

$$\begin{cases} x_{i+1} = 1 + y_i - 1.4x_i^2 \\ y_{i+1} = 0.3x_i \end{cases}, x_0 = 0, y_0 = 0$$

先编写求解方程的函数文件,然后调用该函数文件求 30 000 个点上的 x, y,最后在所有的 (x_i, y_i)坐标处标记一个点(不要连线)绘出图形。这种图形迭代出来的随机点吸引到一起,最后得出貌似连贯的引力线图。

2. 分别利用数值积分法、符号积分法和 Simulink 仿真求 $I = \int_0^1 x\ln(1+x)\mathrm{d}x$。

提示:Simulink 仿真需利用 Simulink 标准模块构建一个积分模型,并把仿真终止时间设置为 1,然后启动仿真,利用 Display 模块显示积分结果。

3. 已知阿波罗(Apollo)卫星的运动轨迹(x, y)满足下列微分方程

$$\ddot{x} = 2\dot{y} + x - \frac{\mu^*(x+\mu)}{r_1^3} - \frac{\mu(x-\mu^*)}{r_2^3}$$

$$\ddot{y} = -2\dot{x} + y - \frac{\mu^* y}{r_1^3} - \frac{\mu y}{r_2^3}$$

其中 $\mu = 1/82.45, \mu^* = 1 - \mu, r_1 = \sqrt{(x+\mu)^2 + y^2}, r_2 = \sqrt{(x+\mu^*)^2 + y^2}$,试在初值 $x(0) = 1.2, \dot{x}(0) = 0, y(0) = 0, \dot{y}(0) = -1.04935751$ 下进行数值求解,并绘制出阿波罗卫星位置(x, y)的轨迹。

提示:先选择一组状态变量,写出一阶常微分方程组,并定义相应的函数文件,然后求方程的数值解。

参考文献

1 刘卫国.科学计算与 MATLAB 语言.北京:中国铁道出版社,2000

2 张志涌.精通 MATLAB 5.3 版.北京:北京航空航天大学出版社,2000

3 薛定宇.反馈控制系统的分析与设计——MATLAB 语言应用.北京:清华大学出版社,2000

4 Redfern D, Campbell C. The MATLAB 5 Handbook. Springer-Verlag, 1998

5 Hanselman D, Littlefield B. Mastering MATLAB 5. Prentice Hall, 1998

6 Biran A, Breiner M. MATLAB 5 for Engineers. Addison-Wesley, 1999

7 Mokhtari M, Marie M. Engineering Applications of MATLAB 5.3 and Simulink 3. Springer-Verlag, 2000

8 Hunt B, Lipsman R, Rosenberg J. A Guide to MATLAB: for Beginners and Experienced Users. Cambridge University Press, 2001

9 Etter D. Engineering Problem Solving with MATLAB, Second edition. Prentice Hall, 1997

10 Herniter M. Programming in MATLAB. Brooks/Cole Publishing Company, 2001

11 Chapman S. MATLAB Programming for Engineers. Brooks/Cole Publishing Company, 2000

12 Eva Part-Enander, Ander Sjoberg 著,王艳清,孙锋,朱群雄等译.MATLAB 5 手册.北京:机械工业出版社,2000

13 许波,刘征.MATLAB 工程数学应用.北京:清华大学出版社,2000

14 陈怀琛.MATLAB 及其在理工课程中的应用指南.西安:西安电子科技大学出版社,2000

15 王华.MATLAB 在电信工程中的应用.北京:中国水利水电出版社,2001

16 蒋尔雄,高坤敏,吴景琨.线性代数.北京:人民教育出版社,1978

17 李荣华,冯果忱.微分方程数值解法.北京:人民教育出版社,1980

18 杨曙,王省富,周肇锡.矢量分析·复变函数·积分变换.北京:国防工业出版社,1980